Computability and Probabilistic machines

K. de Leeuw, E. F. Moore, C. E. Shannon and N. Shapiro
in Automata Studies, Shannon, C. E. and McCarthy, J. Eds. Annals of Mathematics Studies, Princeton University Press

Nigel Phillips

A little context

"*This paper is concerned with finite automata from the experimental point of view. …*
The behavior of these machines is strictly deterministic (i.e., no random elements are permitted in the machine) …
The point of view of this paper might also be extended to probabilistic machines (such as the noisy discrete channel of communication theory), but this will not be attempted here."

Edward F. Moore, (1956) Gedanken-Experiments on Sequential Machines, in Automata Studies, Shannon, C. E. and McCarthy, J. Eds. Annals of Mathematics Studies, Princeton University Press

A little context

"I was particularly impressed by the papers of Kleene and Moore in Automata Studies, and decided to examine the axiomatics of Kleene's algebra and to improve Moore's bounds for length of experiments." (p:vii)

"Any actual machine is finite in the obvious sense that it has but a finite number of component parts … . Any actual machine is also to some extent unreliable, … . We choose to ignore this unsavoury phenomenon, arguing that the deterministic theory is at once simpler and more interesting, and can in any case be made to include probabilistic theory by the simple device of adding new input facilities accessible only to gremlins." (p: 1)

John H Conway, (1971) Regular Algebra and Finite Machines, Chapman Hall.

Caveat

… our results refer to the possibility of enumeration of infinite sets and the computation of infinite sequences. They yield no information of the type that would be wanted if finite tasks were being considered; for example the relative complexity of probabilistic machines that can perform a given finite task and their relative speeds.

Basic Idea

A probabilistic machine is a machine whose input is an infinite series of zeros and ones determined by the output of a binary random device with probability *p* of producing a 1 and that outputs a string using a finite or countably infinite selection of symbols.

Surprising finding

If such a machine's random device has a probability of *p*, where *p* is a computable real number (that is, a real number such that there is an effective process for finding any digit of its binary expansion) any set that can be enumerated by a probabilistic machine can be enumerated by a deterministic machine. This is not true if *p* in a non-computable real number.

p = 0.5 is computable, so any output set that can be computed by a machines whose input is generated by a random device with *p* = 0.5 can be generated by a deterministic machine.

Definition: A machine is a function that, for every n, to each n-tuple of 0's and 1's $(a_1, \ldots, a_n)$, associated with some finite sequence
$f(a_1, \ldots, a_n) = (s_{j1}, \ldots, s_{jr})$ consisting of elements from some fixed set S such that the following two conditions are satisfied.

1. $f(a_1, \ldots, a_n)$ is and initial segment of $f(a_1, \ldots, a_n, \ldots, a_{n+m})$ if $(a_1, \ldots, a_n)$ is an initial segment of $(a_1, \ldots, a_n, \ldots, a_{n+m})$.
2. f is a computable function, that is, there is an effective process such that one can determine $f(a_1, \ldots, a_n)$ if $(a_1, \ldots, a_n)$ is given.

Extending to infinite sequences
If $A = (a_1, a_2, a_3, \ldots)$, f(A) is the sequence which has as initial segments the $f(a_1, \ldots, a_n)$

Example Machine 1

The output symbols of this machine are to be ordered pairs of integers *(a, r)*.

For each input symbol *a* the machine prints the output symbol *(a, r)* if *a* was the r^{th} input symbol on the input tape.

I.e. $f(a_1, \ldots, a_n) = ((a_1, 1), (a_2, 2), \ldots, (a_n, n))$

Example Machine 2

Let *g* be an integral valued function of positive integers which is computable, that's is, there is an effective process for finding *g(n)* if *n* is given. Let the machine print the output symbol (*r, g(r)*) after it has scanned the r^{th} input symbol. In this case

$f(a_1, \ldots, a_n) = ((1, g(1)), (2, g(2)), \ldots (n, g(n))$ – the output is independent of the input.

Example Machine 3

Let the machine print the symbol *0* as soon as it comes to the first zero in the input sequence. Otherwise it is to do nothing. Then $f(a_1, \ldots, a_n) = (0)$ if one of the a_j is a zero. Otherwise the machine prints nothing denoted by *(.)*.

Example Machine 4

Let $f(a_1, \ldots, a_n) = (a_1)$. The machine copies the first input symbol onto the output tape and then prints nothing

Example machine 5

Let the machine print the output symbol *r* if the maximum length of a string of *1s* that the machine has observed is *r*.

E.g. *f(1) = (1),*

f(1, 0) = (1, 1),

f(1, 0, 1) = (1, 1, 1),

f(1, 0, 1, 1) = (1, 1, 1, 2)

Example Machine 6

Let *h(r, s)* be a computable integral valued function whose domain is the positive integers. The machine prints nothing until it come to the first *1* in the input; if this *1* is the first input it never prints anything; if the *1* has occurred after *r* zeros, it prints the symbol *(1, h(r, 1)).* After the next input the machine prints *(2, h(r, 2))*, after the next *(3, h(r, 3))* etc.

Let the function h_r with r fixed be defined by $h_r(s) = h(r, s)$

Then this machine computes the function h_r if it is presented with an initial string of r zeroes followed by a 1.

So *0001110110… -> (.)(.)(.)(1, h(3,1)), (2, h(3, 2)), (3, h(3, 3)), (4, h(3, 4), …*

Example machine 7

This machine is given by $f(a_1, \ldots, a_n) = ((1, r_1), (2, r_2), \ldots, (q, r_q))$ where the integer r_s is obtained as follows:
Look at the first 100^s digits of $(a_1, \ldots, a_n)$.
Let p_s be the proportion of digits that are ones.
Let r_s be the s^{th} digit in the binary expansion of the number p_s. q is the greatest s such that $100^s <= n$.

My interpretation:
Assume for n = 100 (s = 1), p = 0.56
The binary expansion of p is
0.10001111010111000010100011110101110000…
So $r_s = 1$, q = s = 1
So print (1, 1)

Converter

Each machine is defined for a finite input but it is easy to calculated the effect of an infinite input.

Given an infinite tape with only *1s* printed on it the output sequence for machine 1 will be *((1, 1), (1, 2), (1, 3) …)*. Machine 3 gives no output *(.)* and machine 4 *(1)*.

We wish to consider the set of symbols that occur on the output tape of a machine that is supplied with some infinite input sequence A. (For example, if the output sequence is (1, 1, 1, …) the set consists of the symbol 1 alone while if the output sequence is (1, 3, 5, 7, …) the set of symbols is the set of all odd integers.)

Examples

M1 – for any infinite input sequence A = (a_1, a_2, …) the set of output symbols consists of all (a_n, n) as n runs over the positive integers.
M2 – associates any input sequence to the set of all (n, g(n)) as n runs over the positive integers.
M3 – associates the input sequence consisting of all 1s to the empty set.
M5 - associates the input sequence consisting of all 1s to the set of all positive integers

Definition: A machine supplied with the infinite sequence $A = (a_1, a_2, a_3, \ldots)$ will be called an **A-machine**.
It will be said to **A-enumerate** the set of output symbols that it prints.
A set of symbols is called A-enumerable if there is an A-machine that A-enumerates the set.
The sets that are A-enumerable if A is the sequence consisting of all 1s are usually referred to as recursively enumerable sets.
We shall call them **1-enumerable** sets and an A-machine shall be called a **1-machine** if a consists entirely of 1s.

One can associate to each infinite sequence A = (a_1, a_2, …) the set of all pairs (n, a_n) where a_n is the n^{th} element in the sequence A and n runs over all positive integers.

This set will be called A'

A is called computable if there is an effective process by means of which one can determine the n^{th} term of A for every n.

The proofs of the following lemmas are almost immediate.

Lemma 1. The sequence A is computable if and only if the associated set A' is 1-enumerable.

Lemma 2. If A is a computable sequence, any A-enumerable set is 1-enumerable (and conversely).

Lemma 3. If A is not a computable sequence, there exist sets that are A-enumerable but which are not 1-enumerable

LEMMA 1. The sequence ***A*** is computable if and only if the associated set ***A'*** is 1-enumerable.

Proof: Assume that the sequence $A = (a_1, a_2, a_3, \ldots)$ is computable. Define a machine by means of $f(b_1, \ldots, b_n) = ((1, a_1), \ldots, (n, a_n))$. Since ***A*** is computable this actually defines a machine. The machine 1-enumerates the set ***A'***.

To prove the converse, assume that the set ***A'*** is 1-enumerable.

Then there exists a 1-machine whose total output is the set of all symbols (n, a_n), but perhaps not occurring in the proper order.

To determine a_n, the output tape of the machine is observed until a pair whose first entry is the number ***n***. This is certain to eventually happen. The second entry of the pair will be a_n. Thus there is an effective process for determining a_n and ***A*** as a computable sequence

LEMMA 2: If ***A*** is a computable sequence, then any ***A***-enumerable set is ***1***-enumerable (and conversely).

PROOF: Let the machine ***M*** enumerate the set ***S*** if the computable sequence ***A*** = ***(a_1, a_2, a_3, …)*** is used as input. Because of Lemma 1 there is a ***1***-machine that has as output the set of all ***(n, a_n)***.

This output can be converted in an effective manner into the sequence ***(a_1, a_2, a_3, …)*** and used as input by ***M***. The composite object is a ***1***-machine that ***1***-enumerates ***S***.

It is clear that the converse is true, that any ***1***-enumerable set is ***A***-enumerable. This is true because a ***1***-machine that ***1***-enumerates a set ***S*** can be converted into a machine with input, that is oblivious to that input, and will enumerate ***S*** given any input.

LEMMA 3: If ***A*** is not a computable sequence, there exists sets that are ***A***-enumerable but which are not ***1***-enumerable.

PROOF: the set ***A'*** is such a set.

A Random Device

Assume a device that prints ***0***s and ***1***s on a tape, ***1***s occurring with probability ***p*** and ***0***s with probability (1 – ***p***) (0 < p < 1) each printing being independent of the preceding printings. Then the output of the device can be used as the input tape of a machine.

Call such a combination a **p-machine**.

Can any set that can be enumerated by a p-machine by enumerated by a 1-machine?

A set is one enumerable if there is a machine that has the set as its total output if a tape with all 1s is fed in as input. We need a definition of what it means for a set to be p enumerable.

Two definitions are proposed, p-enumerable sets and strongly p-enumerable sets.

Our original informal question will have been reduced to the two precise questions:

Is every p-enumerable set 1-enumerable?
Is every strongly p-enumerable set 1-enumerable?

For a given p-machine one can assign to each set of possible output symbols its probability of occurrence as ? total ? output of that machine.

Assume a ½-machine

- M1 – prints any set with zero probability
- M2 – prints the set of a (n, g(n)) with p=1
- M3 – prints 0 with p=1 but not certainty
- M5 – prints all positive integers p=1 but not certainty
- M6 – prints all (n, h(r, n)) with r fixed ~~$p=2^{-r}$~~ $p=2^{r+1}$

DEFINITION: For some p, a set S of symbols will be called **strongly p-enumerable** if there is a p-machine that produces that set of output symbols in any order with non-zero probability.

From the previous examples the following are strongly p-enumerable:

- The set of all (n, g(n)) (M2).
- The set consisting of 0 alone (M3).
- The set consisting of all positive integers (M5).
- The set of all (n, h(r, n)) with fixed r (M6).

It is clear how a p-machine could be operated to give information about some strongly p-enumerable set. One would operate the p-machine having confidence that the set it enumerates is S in accordance with the probability of S occurring as output. If the probability of S occurring were zero, one would have no confidence in the machine's ability to enumerate precisely S.

However, it might have a high probability of being right for any particular symbol and still enumerate S with zero probability. So that this situation can be considered a weaker definition is given below.

There exists a definite probability that a given p-machine M will eventually print some given output symbol. Let S_M be the set of all output symbols that M eventually will print with probability > ½. For example let the example machines be connected to a random device with p=3/4. Then S_M for
M2 is the set of all (n, g(n))
M3 is the set consisting of 0 alone.
M4 is the set consisting of 1 alone.
M5 is the set of all positive integers.
M6 is the empty set

If a p-machine is started, the probability is > ½ that it will eventually print any given element of S_M, while the probability is <= ½ that it will print any given element not in S_M. Thus it is clear how a machine M could be used to give information about S_M. One would operate the machine having a certain degree of confidence that any particular element of S_M will eventually occur as an output symbol and that any particular element not in S_M will not occur as output.

However, the set S_M itself might have probability 0 in which case one would have no confidence that the machine will enumerate precisely S_M. (It is clear that all this remains true if a number greater than ½ is used in the definition of S_M. All the following results remain valid in this case.)

DEFINITION: A set of symbols S will be called p-enumerable if there is a machine M such that S is precisely S_M.

The condition that a set be p-enumerable is quite weak and we have thought of no weaker definition that would still ensure that there be a p-machine that give some information about each element of the set.
One should note that a p-machine p-enumerates exactly one set while it may strongly p-enumerate many or no sets.
Superficially strong p-enumerability seems a much stronger condition than p-enumerability, and indeed it will be shown that a strongly p-enumerable set is p-enumerable. A bit deeper and perhaps unexpected is the converse result which implies actual equivalence of the two concepts.

One more definition is needed.

Let p be a real number, $0 < p < 1$. The A_p is to be the infinite sequence of 0s and 1s $(a_1, a_2, a_3, \ldots)$ here a_n is the nth digit of the binary expansion of the real number p. That is $p=.a_1a_2a_3\ldots$. Since A_p is an infinite sequence of 0s and 1s, it can be used as input to machines and we shall speak of A_p-machines, A_p-enumerability, etc.

For any number p between 0 and 1, there are now three concepts, Ap-enumerability, p-enumerability and strong p-enumerability. The first is a strictly deterministic notion while the others are probabilistic.

The key result is:

THEOREM 1: Let S be a set of symbols and p a real number, $0 < p < 1$. The following 3 statements are equivalent:

1. S is A_p-enumerable
2. S is p-enumerable
3. S is strongly p-enumerable

This theorem gives immediate answers to out two questions.

First, let p be a non-computable real number between 0 and 1. It is know that such exist and in fact almost all (in Lebesgue measure) numbers are non-computable.

The non-computability of p is equivalent to the none computability of A_p. ???

The according to lemma 3, there exists a set S that is A_p-enumerable that is not 1-enumerable. Thus if one emits random devices that will print the symbol 1 a with probability that is not a computable real number, one can construct p-machines that will “enumerate” sets that are not 1-enumerble.

The situation is entirely different if p is a computable real number (in particular if it is ½).

This is equivalent to A_p being a computable sequence. ???

Theorem 1 shows that any sequence that is p-enumerable or strongly p-enumerable is A_p-enumerable. Since A_p is computable, Lemma 2 shows that the set must be 1-enumerable. Thus a p-enumerable or strongly p-enumerable set must be 1-enumerable.
The conclusion is that if p is restricted to a computable real number (or in particular ½), a machine with a random device could not “enumerate” anything that a deterministic device could not.

Both case can be summarised in:
Theorem 2: If p is a computable real number, any p-enumerable or strongly p-enumerable set is already 1-enumerable. If p is not a computable real number, there exist p-enumerable and strongly p-enumerable sets that are not 1-enumerable

The above seem to me to be nonsense because: If p = ½, then the string of inputs generated will with probability 1 (but not certainty) be a non-computable number. In fact this would be true if p were any real number between 0 and 1.