KYBERNETIKA ČÍSLO 1, ROČNÍK 4/1968

Non-Deterministic Behaviour of Deterministic Computer

JIŘÍ SOUKUP

Current computers are usually regarded as deterministically working machines. The article shows that in circumstances resembling living conditions of live organisms computers can also work non-deterministically. The fundamental theme is illustrated on a program for non-deterministic learning. The article simultaneously touches two fields of questions: problems of design and operation of control systems of living organisms on the one hand, and non-deterministically working programs on the other.

When analyzing the behaviour of living systems we come to the conclusion that there is a certain non-determinateness in their behaviour. So far it is difficult to decide whether it is really non-determinateness, or a sign of their great complexity. It is a fact that in this very non-determinateness one of the fundamental differences between contemporary computers and the control systems of living organisms is seen. Besides this it is known that the range of problems that can be solved by deterministic computers (the Turing machine) is limited and future computers are supposed to be at least partially non-deterministic.

The aim of this article is to show that the non-determinateness of living systems could be concealed in the fact that these systems must react to their surroundings not at the moments which they choose by themselves but at the moments defined by their surroundings, that means randomly from the point of view of the inner rhythm of these systems. This reflection has two significations: 1. Then it would be possible to presume that even living systems working non-deterministically are combined from elements working deterministically (similarly as contemporary computers). 2. This advance would show how to gain non-deterministic behaviour of contemporary computers.

The fundamental idea is illustrated by the program for non-deterministic learning. The program serves for teaching the computer by repeated learing to find the result z to two input signals x, y. The program given below is in Algol 60 with the proceedures of the computer SAAB D 21, which the program has been verified on.

2 TAPECODE (1,2,0,0) is a declaration of the procedure of input and output.

READ (N,1) means a statement to read the variable *N* from the punched tape.

PRINT (≮ **2CR** ANSWER **3SP** ≯, X,2,3) means an order to print 2 empty lines, the text ANSWER, 3 spaces and the values of the variable *X* with 3 decimal places and 2 integer places.

PRINTTEXT has the same meaning as PRINT, but the value of no variable is printed.

YVR has the value of a whole number adjusted binarily on the push buttons of the panel.

The heart of the program is in the part A1, which forms a kind of roulette for the number *K*, stopped at random by an outside impulse (by pushing a button). The other parts of the program only serve to load or read information on the basis of this *K* — see description of activity shown after the program.

PROGRAM

```
begin
comment non-deterministic learning of the deterministic computer;
integer N;
TAPECODE(1,2,0,0);
read (N,1);
begin integer J, K;
    real E. F, G;
    real array X [1 : N], Y [1 : N], Z [1 : N];
    switch S1 := A6, A2, A6, A2;
    switch S2 := A1, A3, A3, A3;
    switch T1 := A1, A1, A4, A4;
    switch T2 := A1, A5, A5, A5;
    K := 1;
A1: if K = N then 1 else K + 1;
    go to S1 [YVR + 1];
A2: read (X [K], 1); read (Y [K], 1); read (Z [K], 1);
    print (≮ cr ≯, X [K], 1, 0);
    print (≮ 2sp ≯, Y [K], 1, 0);
    print (≮ 2sp ≯, Z [K], 1, 0);
    printtext (≮ 2sp learning ≯);
A3: go to S2 [YVR + 1];
A4: read (E, 1); read (F, 1); read (G, 1);
    print (≮ cr ≯, E, 1, 0); print (≮ 2sp ≯, F, 1, 0);
```

```
    for J := 1 step 1 until N do
    begin if K = N then K := 1 else K := K + 1;
          if E = X[K] then begin if F = Y[K] then
                                 begin print(« 2sp », Z[K], 1, 0);
                                       if G noless 0 then
                                       begin if G = Z[K]
    then printtext(« 2sp correct answer »)
    else print(« 2sp bad answer to 2sp », G, 1, 0)
                                 end else
                                 printtext(« 2sp answer only »);
                                 go to A5
                            end
                      end
end;
printtext(« 6sp no answer »);
A5: go to T2[YVR + 1];
A6: go to T1[YVR + 1];                          end end
```

USE OF PROGRAM

The tape containing the number N (maximum number of relations that can theoretically be remembered) is inserted into the tape reader. Then the punched tapes with the triples of numbers X, Y, Z are inserted. If Z is non-negative a process of learning takes place which is directed by the push buttons on the desk like this: If button No. 1 (YVR = 2^0) is depressed and released the phase of learning takes place. If button No. 2 (YVR = 2^1) is depressed and released the computer answers the control question evaluating the result. If Z is negative it means that the answer to the question is not known and the computer answers without regard to Z and without regard to the fact which button was depressed and released (text "answer only").

In the case of repeated learning the punch tape with all the relations X, Y, Z can be spliced into an endless loop.

The behaviour of the program is very similar to the process of learning, for instance learning of human being. As a basis were taken the following properties: 1. The probability of the correct answer increases (in the sense of statistics) with the number of learning cycles. 2. The capacity of the memory is limited. 3. The forgeting follows from the too large demands on the memory or from the fact that the remembering has not been restored by repeating. 4. The loading of the memory, the forgeting and the choice of the answer are provided non-deterministically.

This is illustrated by the notes at the partial results of the example mentioned below. In the case $N = 20$, the computer has learnt to couple with the numbers 1

and 2 their sum correctly by long repetitions. In this situation we began to teach the computer to couple with the numbers 1 and 2 their product like this:

(The reactions of the computer at the first checking.)

1 1 2 incorrect answer to 1
1 2 3 incorrect answer to 2
2 1 3 incorrect answer to 2
2 2 4 correct answer

(The fourth relation happens to coincide. We shall begin to teach.)

1 1 1 learning
1 2 2 learning
2 1 2 learning
1 1 1 learning
1 2 2 learning
2 1 2 learning
2 2 4 learning
1 1 1 correct answer
1 2 2 correct answer
2 1 3 incorrect answer to 2

(still as for adding)

2 2 4 correct answer
1 1 1 learning
1 2 3 incorrect answer to 2

(Although it answered correctly a little while ago. Either it is not sure, i.e. both triples 123 and 122 are in the memory, or it has meanwhile forgotten the triple 122.)

2 1 2 learning
2 2 4 learning
1 2 2 learning
2 1 2 learning
2 2 4 learning
1 1 1 learning
1 2 2 correct answer
2 1 2 correct answer
2 2 4 correct answer
1 1 1 correct answer
1 2 2 correct answer
2 1 3 incorrect answer to 2

(It still is not sure in this answer. Sometimes it answers correctly, sometimes not. Let us try the checking once more.)

2 2 4 correct answer
1 1 1 correct answer
1 2 2 correct answer
2 1 3 incorrect answer to 2

(The same error again. Let us continue in the teaching!)

2 2 4 learning
1 1 1 learning
1 2 2 learning

2 1 2 learning
2 2 4 learning

(Now the computer always answered correctly to repeated questions so that we deduced that it had "learnt" the multiplication.)

PROBLEMS

The given program is designed so that not only the remembering but also the answer are not determinated. But it is not clear whether it would not correspond more with the method of remembering in the human memory (if we should be interested in such a model) for the question to be formulated as a mean (possibly weighted) of all the answers loaded in the memory or the answer that appears in the memory most frequently. As well we could consider the computer reacting to the evaluation of the control question, for instance, in such a way that in the case of an incorrect answer it would delete the relation from its memory or remember it as an incorrect answer. Modifications of the program for such alternatives are simple and we shall therefore not deal with them.

(Received July 17th, 1967)

VÝTAH

Nedeterministické chování deterministického počítače

JIŘÍ SOUKUP

Běžné počítače jsou obvykle považovány za deterministicky pracující stroje. Článek ukazuje, že za podmínek odpovídajících životním podmínkám živých organismů může počítač pracovat nedeterministicky. Základní myšlenka je ilustrována programem pro nedeterministické učení. Uvedené výsledky byly získány na počítači SAAB D 21. Článek se současně dotýká dvou oblastí problémů: problému konstrukce a činnosti řízení v živých organismech na jedné straně a nedeterministicky pracujících programů na straně druhé. To může mít velký význam u různých programů pro učení a u programů, které mají reagovat na nepředpokládané situace v řízených objektech.

Ing. Jiří Soukup, CSc., Výzkumný ústav ekonomiky hornictví, Praha 1, Lazarská 7.