Leonardo, Vol. 14, No. 2, pp. 137–139. 1981.
Printed in Great Britain.
0024–094X/81/020137–03$02.00/0

ARTIFICIAL INTELLIGENCE AND VISUAL ART*

Peter Kugel**

I. INTRODUCTION

In recent years, numerous attempts have been made to use digital computers to produce visual artworks [1] and some to understand their characteristics [1(a)–3]. This may seem rather odd if one thinks of computers as devices that merely do arithmetic. However, computers are widely believed today to be universal symbol-processing systems, and, if this is so, then they should be capable of analyzing artworks.

I share with some of my colleagues in computer science a strong feeling of optimism about what can be learned by trying to produce and to analyze artworks by computer, just as I share a strong feeling of optimism about the capability of computers to deal with other processes carried out by humans. I also believe that the way computers are used today to deal with human thinking can be applied to thoughts of aestheticians and art critics and to the thinking that lies behind at least some kinds of artworks. But those applying computers to the understanding of artworks have imposed on themselves an unnecessary limitation that must be removed or else the understanding will lead to artworks computers produce that are flat and lifeless.

II. WHY SHOULD ARTISTS USE COMPUTERS?

On seeing a bear riding a unicycle, someone once remarked that it was an impressive achievement but wondered why anybody would want to train a bear to do such a thing. One might make the same remark about a computer program that either produced or criticized an artwork. One reason artists might wish to use computers is to produce variants of their artworks in order to relieve themselves of manual labor. Another reason is that computers may help one to obtain a better understanding of the aesthetic qualities of artworks and of the processes by which they are produced. It is this second aim that seems to me to be the main potential merit of computers for artists and aestheticians.

There are at least two ways that a computer program can help one to understand a process. One way occurs when a program is formulated, because writing a program helps one to understand something in very much the way that teaching somebody something helps one to better understand what one is teaching. Programming forces one to become clearer about the factors involved in a process or else computers (which are far dumber in such matters than humans) will fail to understand what they are supposed to do.

The second way is for programs to serve as models of artistic processes—models that can be inspected and tested in ways that actual processes cannot. For example, the cognitive aspects of the art-making process take place in an artist's brain and at present they are not available for inspection by others, whereas a computer program for an artwork is written on a piece of paper, and the aspects are clearly stated.

III. BRAIN, MIND AND THINKING

What is one to do if one wants to develop a precise account of thinking? One approach has been offered by behaviorist psychologists. They simplified matters by suggesting that thinking operates like the process diagrammed in Fig. 1. Since one cannot at present obtain information on what goes on inside the brain, argued behaviorists, one should analyze only the information input to the brain and the human response or output.

But one cannot help asking, what goes on between the inputs and outputs? What happens in the brain in the middle? Neurologists may point to an open cranium of a cadaver and say: Look, here it is, the thing that thinks. And, if one then replies that one cannot see anything but peculiar convolutions, such neurologists may cut out a slice of the brain and place it under a microscope so that one can better see details.

It is little wonder that many people, confronted with such approaches, turn to computer scientists who say: You want to know what goes on inside the human brain? Then try to design a machine capable of doing what a brain does. At least, one can inspect the machine in order to understand how it functions. It may work in the same way as the brain does, but, if it does not, then at least one has a machine that may have other uses.

IV. CAN DIGITAL COMPUTERS THINK?

When in science one tries to understand something, say the way objects fall to the Earth under the force of gravity,

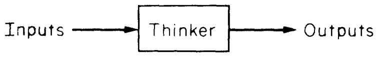

Fig. 1. Diagram of a thinker

*Parts of this article are taken from The Controversy Goes On: Can Computers Think?, which appeared in *Creative Computing*, Nos. 8, 9 and 10, 1979. Permission given by the publisher, *Creative Computing*, P.O. Box 789-M, Morristown, NJ, U.S.A.

**Computer scientist, Boston College, Chesnut Hill, MA 02167, U.S.A.

one proposes a model for the process. If the model can be formulated in precise form, it can be programmed for a computer and tested by the computer. Similarly, if one can formulate a precise model of thinking in general, or of artistic thinking in particular, one should be able to program it and test it by computer.

Attempts have been made in recent centuries to build, or at least to design, 'thinking' machines. Current interest in such machines derives from the availability of the digital computer, which is different in kind from machines that have been considered in the past.

If any machine can be constructed to 'think' or be capable of what is called Artificial Intelligence (AI), it is now believed that the digital computer is the machine. This follows from a thesis suggested by the British mathematician A. M. Turing [4], according to whom that which can be done by any machine by way of symbol processing can be done by computer.

Turing asked himself an important question: How many different types of computing machines must be designed if all kinds of information processing are to be carried out mechanically? His answer was that it is possible to design a *single* machine that merely requires different programs and this is called a *Universal Turing Machine*. Present-day digital computers theoretically can meet this condition. From this it follows that if digital computers cannot be programmed to behave intelligently or to provide AI, then AI cannot be achieved by any machine.

If thinking cannot be carried out mechanically, thinking must involve non-mechanical processes. Some computer scientists do not accept this possibility, since they feel this would imply that thinking cannot be treated scientifically. If thinking is a mechanical process, then it should be possible to devise a program that causes a computer to think.

Turing [5] suggested a behavioral definition of *thinking*. A machine, stated Turing, can be said to *think* if it can produce the behavior of a thinking person limited to what can be transmitted by means of a teletypewriter. It has been objected that this definition does not take account of what happens inside the brain. But, if a computer can be programmed to satisfy Turing's definition, then an important step forward would be made in the development of AI.

Although humans are not programmed as computers are, they seem to be directed by something that behaves much like a program. Thus, for example, when one learns to play a game such as chess, what one has learned to do is almost certainly represented by some sort of finite configuration in one's mind that can be applied to a particular position in chess in order to make the next move. This configuration guides the mind in evaluating positions in order to make a good next move and the configuration is undoubtedly a lot like a computer program. One of the striking successes of AI research has been the development of chess programs capable of defeating good chess players.

Computer chess programs capable of defeating me might be said to be more intelligent in chess than I am. But I could argue that the computer was not intelligent but merely followed the instructions in programs prepared by more intelligent chess players than I am.

V. CAN COMPUTERS BE TAUGHT TO THINK?

Consider the task of continuing a sequence of numbers when only a few initial numbers are given. Suppose the initial numbers are 2, 4, 6, 8 ... The continuation might be 10, 12, 14, ... Following an idea of Solomonoff [6], one can say that this kind of task is typical of making a generalization from specific examples, a kind of task that also includes the following: (1) A child is given an initial set of utterances in a natural language and learns the basis of the language. (2) A scientist is given specific observations and develops a hypothesis generalizing them in order to predict new observations. (3) You are asked on the basis of these above two examples to develop a concept of learning general ideas from examples.

There are two steps involved in such a process (Fig. 2)—rule formation (Part 1) and rule application (Part 2).

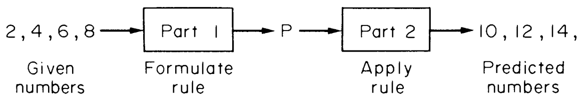

Fig. 2. Diagram of an intelligent system

Computers can apply rules as well as, and sometimes better than, a human. What computers at present do badly is formulate the rules to be applied. One reason is that a given sequence such as 2, 4, 6, 8 ... might be continued by 6, 4, 2, 0, −2, −4, −6 ... or by 8, 8, 8 .... How does one decide which is the correct continuation? This problem arises in those cases where one is asked to go beyond the information provided. One cannot be sure that a hypothesis based on the information provided will not be invalidated by a new bit of information. The English philosopher David Hume noticed this and thought it made the validity of generalization by induction very doubtful.

But Hume was wrong. Recent developments in mathematical logic [7, 8, 9] suggest that there are processes that can 'go beyond the information given' both with certainty and in finite time. For example, suppose that one wishes to determine if there is an underlying pattern in an infinite sequence of integers. Evidently, one can consider only a small part of such a sequence. This can be done by the following kind of *trial-and-error procedure*.

Consider just enough of the sequence of integers to permit one to formulate a hypothetical rule for a pattern underlying the sequence. Test the rule against more integers of the sequence. If the rule is violated, formulate a new hypothetical rule and repeat the procedure. Now, it is possible that such a procedure will lead, in finite time, to a rule that is not violated, and, if it does, the rule must be correct for the whole infinite sequence of integers.

This kind of procedure, however, cannot be carried out by a digital computer when it is used as a Turing Machine, because such a machine must put out a final rule and stop. A trial-and-error machine cannot stop, because it cannot know when it has put out its final rule.

Computing procedures reach conclusions in finite time, and one can tell when they have done so. This is because one counts the first result they produce. Trial-and-error procedures also reach conclusions in finite time, but one cannot tell when they have done so. This is because one counts the last result they produce (rather than the first). This may seem like a subtle difference, but it makes them much more powerful than computing procedures.

It also makes them a nuisance for computer operators. One cannot turn off a computer running a trial-and-error procedure, because one never knows when it is 'done'. Thus, one cannot deliver the results to a customer, because one cannot know, at any moment, that the procedure has been finished. Trial-and-error procedures may not be useful on a computer, but they can be used to formulate models of the ways humans process information when 'going beyond the information given'. This (I claim) is what happens when people make and analyze artworks, and this is why I think that trial-and-error procedures are necessary for any full model of these kinds of human activities.

The trial-and-error procedure is *open* in much the same way as some human thinking might be said to be open. Persons say they *know* rather than *think* when they arrive at a conclusion and close their minds to other possible conclusions. Computers that print out the form of a rule for the example above of an infinite sequence of integers and then stop may be said to *know* the rule whereas people who behave like a trial-and-error procedure may only say that they *think* they know the rule. Computers that are used to carry out computing procedures can be said to *know*, while, at any moment of time, computers that carry out trial-and-error procedures can only be said to *think*.

Knowing may be better when it comes to doing a payroll. But thinking is better than knowing when one is developing a scientific theory or an artwork.

VI. CAN ARTIFICIAL INTELLIGENCE MACHINES BE DEVELOPED?

Consider a hierarchy of procedures, each of which is more complicated and capable of dealing with more complex problems than those lower in the hierarchy. At the bottom of the hierarchy are the regular procedures used in such finite machines as the thermostat, cuckoo clock and telephone switchboard. Above these are the computing procedures used in computers. Above these are the trial-and-error procedures. Each category in the hierarchy is easier to understand than the one above it, and each of the categories can be used to obtain insights into human thinking. The digital computer, I believe, has provided more such insights during the past 30 years than any other approach, but it leaves things out. The trial-and-error procedure includes more of them, but these are more difficult to understand. However, probably people are more difficult to understand than computers. This may be because people use trial-and-error procedures and computers do not. For other accounts of the differences see Ref. 10.

VII. CONCLUSION

The merit of using a trial-and-error procedure as a model for human thinking and art making, is simply that it allows one to deal with some things that the use of computing procedures as models does not allow. Trial-and-error models in aesthetics can thus characterize more things than computing models can—more things, but not everything.

REFERENCES

1. (a) F. J. Malina, ed., *Visual Art, Mathematics and Computers* (Oxford: Pergamon Press, 1979) and (b) R. Leavitt, ed., *Artist and Computers* (New York: Harmony Books, 1976).
2. M. S. Apter, Can Computers Be Programmed to Appreciate Art, *Leonardo* **10**, 17 (1977) and H. W. Franke, A Cybernetic Approach to Aesthetics, *Leonardo* **10**, 203 (1977).
3. G. Stiny and J. Gips, *Algorithmic Aesthetics: Computer Models for Criticism and Design in the Arts* (Berkeley, Calif.: Univ. California Press, 1978).
4. A. M. Turing, On Computable Numbers with an Application to the Entscheidungsproblem, *Proc. London Math. Soc.*, Ser. 2, **42**, 230 (1936) and A. Church, A Note on the Entscheidungsproblem, *J. Symbolic Logic* **1**, 40 (1936).
5. A. M. Turing, Computing Machinery and Intelligence, *Mind* **59**, 433 (1950).
6. R. J. Solomonoff, A Formal Theory of Inductive Inference, *Information and Control* **7**, 234 (1964).
7. H. Putman, Trial and Error Predicates and the Solution of a Problem of Mostowski, *J. Symbolic Logic* **20**, 49 (1965).
8. E. M. Gold, Limiting Recursion, *J. Symbolic Logic* **20**, 28 (1965) and Language Identification in the Limit, *Information and Control* **10**, 447 (1967).
9. P. Kugel, Induction, Pure and Simple, *Information and Control* **35**, 276 (1977).
10. H. L. Dreyfus, What Computers Still Can't Do, *Creative Computing*, p. 54 (Jan. 1980) and Why Computers Can't Be Intelligent, *Creative Computing*, p. 72 (March 1980).