

## Strong AI and the Problem of “Second-Order” Algorithms

Gerd Gigerenzer

Department of Psychology, University of Constance, Germany

“In my childhood we were always assured that the brain was a telephone switchboard (“What else could it be?”),” recalls John Searle (1984, p. 44). Children today are likely to be told that the mind is a computer program. Roger Penrose, slipping back into the role of the child who dares to question, rejects the “strong AI” claim that “mental activity is simply the carrying out of some well-defined sequence of operations, frequently referred to as an *algorithm*” (p. 17). Penrose argues “that the decision as to the validity of an algorithm is *not* itself an algorithmic process!” (p. 414). Let us call these hypothetical algorithm-checking algorithms, “second-order” algorithms. Penrose cites Turing’s proof that no algorithm exists for deciding the question of whether or not Turing machines will actually stop (i.e., whether algorithms will actually work). In this comment, I will add several thoughts of a more pointedly psychological sort that support Penrose’s mathematical argument.

*Scientific inference.* Inference in science (e.g., from data to hypothesis) is a mental activity in which algorithms actually exist. Various statistical (e.g., Bayesian, Fisherian, Neyman-Pearsonian) and nonstatistical (e.g., Platt’s strong inference) algorithms have been proposed. As is well known, there is little consensus among philosophers, probabilists, and scientists as to which (formal) algorithm applies to which type of (semantic) problem, or whether to use a statistical algorithm at all (Gigerenzer et al., 1989). (There are, however, such “rituals” as the mechanical null hypothesis testing that goes on in some social sciences.) That is, algorithms for scientific inference exist, but there is no “second-order” algorithm for choosing among them. The basic reason for this disagreement is that the problem of inductive inference has no single solution that commands consensus—it has many, competing ones. Indeed, there is no agreement as to whether the problem has a single solution (even in principle). In our current (and perhaps permanent) state of controversy over this question, there is no algorithm for choosing among algorithms—but scientists nonetheless do somehow choose, and with considerable success. Nor are our choices merely blunt expressions of taste—you like Neyman/Pearson and vanilla, I like Fisher and chocolate, who knows why? We argue with one another, offer reasons for our choices, and sometimes even persuade one another.

*Concept ambiguity.* An algorithm (a Turing machine) is purely syntactical: It specifies, for instance, that if a machine is in a certain state and has a certain symbol on its tape, then the machine will perform a certain operation such as erasing a symbol on the tape and enter another state. The mind, however, has a semantics, too. In many problems (ones that do not deal with well-defined artifacts) that the mind has to handle, there is no simple one-to-one correspondence between a formal concept and a semantic concept. Here, ambiguity first has to be resolved before an algorithm can be put to work—and such judgments depend heavily on content and context rather than on formal structure. Can a formal algorithm resolve this ambiguity in the way humans do?

Consider a judge who is a Bayesian and computes the probability that a suspect actually committed a crime by an algorithm known as Bayes’ rule. One formal concept in this algorithm is the suspect’s *prior probability* (of having committed the crime in question), which needs to be semantically interpreted in each individual case. The ambiguity is not only in the precise number of that probability, but in the *kind of reference class* from which this probability should be taken. Each suspect is always a member of many (usually, an *infinite* number of) reference classes (e.g., single parents, young urban professionals, weight lifters)—and all of them may have widely divergent prior probabilities. From time to time, new, never before thought of reference classes may emerge—for example, after the discovery of a new drug whose use is correlated with a certain kind of crime. Although our Bayesian judge’s reasoning contains an algorithm, as we assumed, the judge also has to assess relevance: which reference class to choose, and which others to ignore. It is hard to see how these judgments can be made mechanically by a “second-order” algorithm.

*Structural ambiguity.* Probabilistic algorithms are based on several structural assumptions (e.g., independence) that must hold in the relevant part of the real world if the algorithm is to be applied validly. Textbook applications, such as “urns-and-marbles” problems, are contrived so that there is a one-to-one correspondence between the structural assumptions of an algorithm and the structure of the problem at hand. Beyond textbook problems, however, we must confront ambiguity about structural correspondence (Gigerenzer & Murray, 1987, chap. 5). Consider the following stories that illustrate how important it is for the mind to check structural assumptions and resolve this ambiguity.

(1) You live in Palo Alto. Today you must choose between two alternatives: to buy a BMW or a Jaguar. You use only one criterion for that choice, the car’s life expectancy. You have information from a test sample of 100 BMWs and 100 Jaguars, of which 75% and 50%, respectively, lasted longer than 10 years. Just yesterday your neighbor told you that her new BMW broke down. Nevertheless, in your reasoning, your neighbor’s case decreases the BMW’s prior probability only slightly, from .75 to about .74. So you go ahead and buy a BMW.

It is easy to specify an algorithm for this kind of decision-making. Now look at the same problem, but with a different content.

(2) You live in a jungle. Today you must choose between two alternatives: to let your child swim in the river, or to let it climb trees instead. You use only one criterion for that choice, your child’s life expectancy. You have information that in the last 100 years there was only one accident in the river, in which a child was eaten by a crocodile, whereas a dozen children have been killed by falling from trees. Just yesterday your neighbor told you that her child was eaten by a crocodile.

If, in your reasoning, the same algorithm is applied again, your neighbor’s testimony would make little difference: The prior probability of a fatal accident in the river would increase only slightly, from one to two cases in 100 years. The algorithmic mind would probably send the child to the river. The mind of a parent, however, might use the new information to *reject* the old algorithm, rather than to *apply* the old algorithm to the new information. The parental mind may suspect that the small river world has changed—crocodiles may now inhabit the river. The updating of prior probabilities may no longer make sense, since the events (being eaten or not) can no longer be considered as independent random drawings from the same reference class. A structural assumption of the algorithm no longer seems to hold. From now on, many children may be eaten.

I do not know of any “second-order” algorithm that is capable of performing this checking of structural assumptions of algorithms in the same way the human mind does and with similar ease. Can an algorithm be sufficient to judge whether one and the same information (your

neighbor's report) is to be interpreted as an *entry* to a computation or as a *rejection* of exactly the same computation? Even for this simplistic structural problem—two alternatives, only one criterion—there seems to be no general algorithm that can compute for all possible contents (and there are *infinitely* many more besides cars and crocodiles) whether the mind uses the prior probability updating algorithm or not. Nevertheless, in individual cases, we may well be able to make an unequivocal judgment. This situation is analogous to the Turing proof: There is no general algorithm that can compute whether algorithms ever stop, but in the individual case we can immediately “see” the answer.

Throughout this discussion I accepted Penrose's view that a large part of human thinking is indeed algorithmic, and added some psychological reflections to his argument that the decision as to the validity of an algorithm is at least in part, nonalgorithmic. This is not to say that I do believe that most human thinking, be it “second-order” or “first-order,” *is* solely algorithmic. Even if the *result* of thinking can be simulated by an algorithm, this does not imply that the *process* of thinking is algorithmic, as John Searle has repeatedly pointed out. If there is a computer algorithm that simulates perfectly the time shown by a mechanical clock, this does not imply that the mechanism by which the clock quantifies time is indeed computing. And whereas AI workers can be content with applications that work—a computer that tells time—we psychologists are still responsible for taking apart the ticking clock.

### *Acknowledgment*

This comment was written while I was a Fellow at the Center for Advanced Study in the Behavioral Sciences, Stanford, CA. I am grateful for financial support provided by the Spencer Foundation and by the Deutsche Forschungsgemeinschaft (DFG). I would like to thank Lorraine Daston and Kathleen Much for their helpful suggestions on this comment.