AI Systems Are Dumb Because AI Researchers Are Too Clever

JACQUES PITRAT

CNRS-LAFORIA, IBP-Université Pierre et Marie Curie, Paris
⟨pitrat@laforia.ibp.fr⟩

The successes of many AI systems are due not to their own intelligence but to the intelligence of their designers. He or she gives them a method to solve the given problem or instructs them how to perform an efficient combinatorial search for this problem: he or she spoonfeeds them. As these systems are unable to find themselves a good method for other problems, they lack generality, an essential part of intelligence. Is it possible to call such systems "intelligent," for instance a chess program that defeats the world champion but is unable to play any other game?

This limitation results from the inability of most systems to work at two levels. They do not access the formulation of problems, they only know what they can do in some situations. For instance, they wander in a maze which they explore systematically. However, people may be efficient general problem solvers. At the upper level (or metalevel), they examine the formulation of the problem or the behavior of the system; at the lower level, they solve the problem. Intelligence entails, and is only possible through, such metareasoning; unfortunately, metareasoning in AI systems is usually performed by the designer.

Let us consider the following problem: a snail climbs along a pole 15 meters high. It climbs three meters each day and night it goes down two meters again. When will it reach the top of the pole? Most people immediately answer, "15 days." In the present case, they are wrong, because at the end of the thirteenth day, the snail reaches the top of the pole. But this mistake shows that they misused a generally useful method. From the statement of the problem, they were right to state that the snail was 3-2 = 1 meter higher each day. Thus they have proved a metatheorem and they use it to solve the problem: because the pole is 15 meters high, they conclude that 15 days are needed. With a slight modification, we easily find the right answer. We could save a lot of time with the preceding metatheorem if the pole were a billion meters high: instead of performing a billion steps, one finds the solution directly after a short study of this formulation of the problem.

One would like to be able to describe problems to an intelligent system and then let it solve them; after all, we are not supposed to do the job for it! In most systems, the designer has made the intelligent part when he or she worked at the metalevel, defining an efficient method to solve a particular problem. It only remains for the system to do the stupid part: to apply this method.

It is a commonplace to contrast efficiency and generality. The performance of most general systems has certainly been unsatisfactory. On the other hand, when we want to design a system solving only one kind of problem, we inspect the formulation of this problem and prove some results that are used to improve the performance of our system. For instance, in chess we know we must move our king when there is a double check; it is useless to consider any other move. We cannot include such results in a general system, they are true only in a particular domain. If the system cannot examine the formulation of the rules of a game, it cannot simultaneously efficient and general. The results, specific to each game, that could improve the search are lacking. For that reason, general systems have a bad reputation.

But a system that works at two levels can study the formulation of each particular problem to infer new methods in order to solve the problem more easily. For instance, on the metalevel it examines the structure of a maze so as to find its laws and discover useful shortcuts by itself. At the lower level, it selects its actions according to the principles it found at the metalevel. So such a general system performs as well in every domain as a system specifically designed for this domain. For example, a general theorem prover [Pitrat 1970] builds for each theory a set of new ways to prove theorems, which are new metatheorems. Thanks to them, it can more easily find proofs that are more elegant than those obtained with only the initial set of tools. Such systems are both general and efficient.

A general system that can study the formulation of problems may be even more successful than a program specifically written to solve only one problem. This paradox can be explained easily. The designer of a general system adds a new method when it is obviously useful for some particular problem. Once the new method is in the set of methods, it can be used by this general system for any problem: it chooses among all methods the one it will use. Sometimes, the system is right to choose at the metalevel a method that people do not think of using for this kind of problems, so that programmers do not include it in a system that only solves this problem. Without this method, however, this specific system cannot be as efficient as the general one.

There are many other reasons to devise systems able to work at the metalevel. For instance, an autonomous systems should analyze at this level what it has done at the lower level and compare the results with what was expected beforehand. Such a system monitors the search of the solution and avoids hopeless wandering. Moreover, it is also more profitable to give a system metaknowledge than knowledge. To give knowledge is like giving a fish to a starving man, to give metaknowledge is like teaching him how to fish. It is a colossal task to provide various systems with the knowledge they need. Let us rather give a general system metaknowledge to find knowledge useful for each particular problem.

An AI system unable to work at two levels cannot be at the same time general, successful and autonomous. It does not deserve to be called "intelligent."

REFERENCES

PITRAT, J 1990. *Métaconnaissance, Future de l'Intelligence Artificielle*, Hermès.

PITRAT, J 1970. Heuristic interest of using metatheorems. In *Lecture Notes in Mathematics 125*, Springer-Verlag, 194–206.