Knowledge Is Power

Michael Genesereth
Computer Science Department
Stanford University

Personal perspective on Logic-based Artificial Intelligence requested by Illah Nourbakhsh and Michael Druckman for inclusion in *Crunch Time: Career Pathways in the Age of AI*

> "Knowledge is power, and the computer is an amplifier of that power. We are now at the dawn of a new computer revolution ... Knowledge itself is to become the new wealth of nations." - Ed Feigenbaum, "The Fifth Generation" 1983

A common theme in early work on AI was the importance of knowledge to intelligent behavior. Researchers by and large agreed that intelligent systems must possess adequate knowledge of their application areas in order to succeed, while systems with inadequate knowledge are unlikely to achieve their goals. Despite this general agreement, there was significant debate about the representation of that knowledge.

Some researchers advocated implicit representations. They argued that knowledge does not need to be represented explicitly in order for systems to fulfill their goals. A thermostat senses temperature and supplies power to the house's furnace, but does it *know* that doing so will change the house's temperature? A leaf turns toward the sun and thereby increases food production, but does it *know* that is the reason for phototropism? A sorting program orders lists of numbers, but does it *know* that the "less than" relation is transitive?

Other researchers, myself included, advocated "declarative" representations (in which knowledge and goals are represented explicitly, typically encoded as sentences in the language of formal logic). Declarative knowledge representation has multiple advantages. It makes systems easier to build and easier to modify (since there is no programming in the traditional sense). It makes systems more versatile (able to achieve multiple goals in one environment and able to achieve the same goal in multiple environments). And it makes it possible for systems to explain their results and their actions in terms that are understandable to their users, something that thermostats and tree leaves and sorting programs do not do.

John McCarthy (the father of AI) argued strongly for a logic-based representation of knowledge. His idea was simple. He wanted a machine that could be "programmed" by description. The user would write sentences describing the application environment; he would write sentences describing the desired goal; and the machine would use that information in determining its behavior. There would be no programming in the traditional sense. McCarthy presented his concept (called the "advice taker") in a paper that is now a classic in the field.

> "The main advantage we expect the advice taker to have is that its behavior will be improvable merely by making statements to it, telling it about its ... environment and what is wanted from it."

An ambitious goal! But that was a time of high hopes and grand ambitions. The idea was echoed by subsequent researchers. In a paper written in 1974, Feigenbaum, the inventor of Knowledge Engineering, described his version of McCarthy's ideal.

> "The potential use of computers by people to accomplish tasks can be "one-dimensionalized" into a spectrum representing the nature of the instruction that must be given the computer to do its job. Call it the what-to-how spectrum. At one extreme of the spectrum, the user supplies his intelligence to instruct the machine with precision exactly how to do his job step-by-step. ... At the other end of the spectrum is the user with his real problem. ... He aspires to communicate what he wants done ... without having to lay out in detail all necessary subgoals for adequate performance."

Bob Kowalski, the co-inventor of "logic programming", pointed out the superior versatility of logic programs over traditional programs.

> "We know that a grandparent is the parent of a parent. That single definition can be used for multiple purposes. (1) We can use it to compute whether one person is the grandparent of a second person. (2) We can use the definition to compute a person's grandparents. (3) We can use it to compute the grandchildren of a given person. (4) And we can use it to compute a table of grandparents and grandchildren. In traditional programming, one would write different programs for each of these tasks, and the definition of grandparent would not be explicitly encoded in any of these programs. In "logic programming", the definition would be written just once, and that single definition could be used to accomplish all four tasks."

Of course, just having knowledge is not sufficient to ensure intelligent behavior. A system must also be able to "operationalize" that knowledge to answer questions and/or achieve its goals. To this end, early researchers in AI devoted significant effort to automated reasoning, with emphasis on logical deduction, leading to the development of powerful automated reasoning systems (such as resolution and Prover9 and Vampire). The need to operationalize knowledge to achieve goals in dynamic worlds led to progress on automated planning, automatic programming (e.g. the seminal work of Cordell Green and Richard Waldinger), and game playing.

An especially intriguing idea inspired by the prospect of automated reasoning systems and automated planning systems was the concept of computer systems that are able to reason about their own cognition (metareasoning) and that can program themselves (metaprogramming).

The General Problem Solver (GPS) developed by Newell and Simon transitioned to metareasoning when it encountered a problem solving "impasse" in ordinary reasoning. Subsequently, Weyhrauch and others investigated ways in which "reflection" on a problem could be used by a system to solve "the problem of solving a problem" and thereby achieve advantages in efficiency.

In the early 2000s, General Game Playing (GGP) emerged as a testbed for metaprogramming systems. As with traditional game playing, success at GGP involves game tree search. However, successful general game players also rely on a form of self-programming called metagaming, i.e. reasoning about a game's rules in order to create programs to play that game. Game descriptions are typically much smaller than game trees, and the cost of metagaming is often proportional to the size of the description rather than the size of the game tree. In such cases, players can expend a little time in game analysis and gain disproportioniate savings in game tree search. Luckily, this has been borne out in practice. Using traditional game playing techniques together with metagaming, students are able to create programs running on laptops that excel in GGP competitions and that routinely beat human beings at games that neither have seen before.

For many early researchers, the ultimate goal of AI was to build autonomous systems that can rival human intellectual capabilities (and pass the Turing test). Others, myself included, wondered why these folks would set their sights *so low*. If we can build bulldozers that outperform humans and calculators that can add faster, why not build *superintelligent* systems that can outthink humans? Proponents of this position argued that, given enough knowledge, the ability to reason with that knowledge, and the ability to program themselves, such systems could far exceed human performance.

But there is more. For many early researchers, the ability to perform well on specific tasks was the hallmark of intelligence. For others, myself included, *generality* was deemed to be equally or even more important. We wanted systems that are general enough to solve problems they have never seen before, based only on knowledge that, like general game players, is received "at runtime".

> To quote Robert Heinlein: "A human should be able to change a diaper, plan an invasion, butcher a hog, conn a ship, design a building, write a sonnet, balance accounts, build a wall, set a bone, comfort the dying, take orders, give orders, cooperate, act alone, solve equations, analyze a new problem, pitch manure, program a computer, cook a tasty meal, fight efficiently, die gallantly. Specialization is for insects." The same holds for AI systems.

As one whose career has spanned decades from the earliest work on AI to the present, the prospect of such systems is what gets me up in the morning today. (1) I want systems like the ones that McCarthy recommended - ones that are "improvable" merely by making statements to them, telling them about their ... environment and what is wanted. (2) I want systems that are *versatile* and do not require me to tell them how to do everything. (3) I want systems that are *reliable*. (4) I want systems that can *explain* their actions. I would like to see superintelligences, systems that are capable of high performance on familiar problems and at the same time are general, i.e. able to solve problems no one has seen before.

Although we are not there yet, I am not frustrated by the work thus far or the problems that are yet to be solved. My frustration comes from the popularity of superficial approximations to the capabilities just described. I am frustrated that funders and the press and the public are fixated on technologies that give the appearance of intelligence despite having little real understanding. Although these technologies do have uses, they are not candidates for artificial intelligence. As far as research in AI is concerned, they may even be harmful in that they reinforce the popular view that machines cannot be truly intelligent.

To a great extent, the progress of the human race has come about because of increases in our knowledge and our use of that knowledge in developing beneficial technologies (e.g. agriculture, calculators, autopilots, medicines, and so forth). A similar argument can be made for the value of knowledge and knowledge technology to the computer-based systems of the future and the potential benefits of those systems for humanity.