KurzweilAI.net

Why We Can Be Confident of Turing Test Capability Within a Quarter Century

by Ray Kurzweil

The advent of strong AI (exceeding human intelligence) is the most important transformation this century will see, and it will happen within 25 years, says Ray Kurzweil, who will present this paper at The Dartmouth Artificial Intelligence Conference: The next 50 years (AI@50) on July 14, 2006.

Ray Kurzweil will present this paper at The Dartmouth Artificial Intelligence Conference: The next 50 years (AI@50) in Hanover, NH on Friday July 14, 2006 in a talk, "Why We Can Be Confident of Turing Test Capability Within a Quarter Century."

Published on KurzweilAI.net July 13, 2006. Excerpted from The Singularity is Near, When Humans Transcend Biology *by Ray Kurzweil.*

> *Consider another argument put forth by Turing. So far we have constructed only fairly simple and predictable artifacts. When we increase the complexity of our machines, there may, perhaps, be surprises in store for us. He draws a parallel with a fission pile. Below a certain "critical" size, nothing much happens: but above the critical size, the sparks begin to fly. So too, perhaps, with brains and machines. Most brains and all machines are, at present "sub-critical"—they react to incoming stimuli in a stodgy and uninteresting way, have no ideas of their own, can produce only stock responses—but a few brains at present, and possibly some machines in the future, are super-critical, and scintillate on their own account. Turing is suggesting that it is only a matter of complexity, and that above a certain level of complexity a qualitative difference appears, so that "super-critical" machines will be quite unlike the simple ones hitherto envisaged.* —J. R. Lucas, Oxford philosopher, in his 1961 essay "Minds, Machines, and Gödel"[1]

> *Given that superintelligence will one day be technologically feasible, will people choose to develop it? This question can pretty confidently be answered in the affirmative. Associated with every step along the road to superintelligence are enormous economic payoffs. The computer industry invests huge sums in the next generation of hardware and software, and it will continue doing so as long as there is a competitive pressure and profits to be made. People want better computers and smarter software, and they want the benefits these machines can help produce. Better medical drugs; relief for humans from the need to perform boring or dangerous jobs; entertainment—there is no end to the list of consumer-benefits. There is also a strong military motive to develop*

> *artificial intelligence. And nowhere on the path is there any natural stopping point where technophobics could plausibly argue "hither but not further.* —Nick Bostrom, "How Long Before Superintelligence?" 1997
>
> *It is hard to think of any problem that a superintelligence could not either solve or at least help us solve. Disease, poverty, environmental destruction, unnecessary suffering of all kinds: these are things that a superintelligence equipped with advanced nanotechnology would be capable of eliminating. Additionally, a superintelligence could give us indefinite lifespan, either by stopping and reversing the aging process through the use of nanomedicine, or by offering us the option to upload ourselves. A superintelligence could also create opportunities for us to vastly increase our own intellectual and emotional capabilities, and it could assist us in creating a highly appealing experiential world in which we could live lives devoted to joyful game-playing, relating to each other, experiencing, personal growth, and to living closer to our ideals* —Nick Bostrom, "Ethical Issues in Advanced Artificial Intelligence," 2003
>
> *Will robots inherit the earth? Yes, but they will be our children.* —Marvin Minsky, 1995

Of the three primary revolutions underlying the Singularity (G, N, and R), the most profound is R, which refers to the creation of nonbiological intelligence that exceeds that of unenhanced humans. A more intelligent process will inherently outcompete one that is less intelligent, making intelligence the most powerful force in the universe.

While the "R" in GNR stands for robotics, the real issue involved here is strong AI (artificial intelligence that exceeds human intelligence). The standard reason for emphasizing robotics in this formulation is that intelligence needs an embodiment, a physical presence, to affect the world. I disagree with the emphasis on physical presence, however, for I believe that the central concern is intelligence. Intelligence will inherently find a way to influence the world, including creating its own means for embodiment and physical manipulation. Furthermore, we can include physical skills as a fundamental part of intelligence; a large portion of the human brain (the cerebellum, comprising more than half our neurons), for example, is devoted to coordinating our skills and muscles.

Artificial intelligence at human levels will necessarily greatly exceed human intelligence for several reasons. As I pointed out earlier machines can readily share their knowledge. As unenhanced humans we do not have the means of sharing the vast patterns of interneuronal connections and neurotransmitter-concentration levels that comprise our learning, knowledge, and skills, other than through slow, language-based communication. Of course, even this method of communication has been very beneficial, as it has distinguished us from other animals and has been an enabling factor in the creation of technology.

Human skills are able to develop only in ways that have been evolutionarily encouraged. Those skills, which are primarily based on massively parallel pattern recognition, provide proficiency for certain

tasks, such as distinguishing faces, identifying objects, and recognizing language sounds. But they're not suited for many others, such as determining patterns in financial data. Once we fully master pattern-recognition paradigms, machine methods can apply these techniques to any type of pattern.[2]

Machines can pool their resources in ways that humans cannot. Although teams of humans can accomplish both physical and mental feats that individual humans cannot achieve, machines can more easily and readily aggregate their computational, memory and communications resources. As discussed earlier, the Internet is evolving into a worldwide grid of computing resources that can be instantly brought together to form massive supercomputers.

Machines have exacting memories. Contemporary computers can master billions of facts accurately, a capability that is doubling every year.[3] The underlying speed and price-performance of computing itself is doubling every year, and the rate of doubling is itself accelerating.

As human knowledge migrates to the Web, machines will be able to read, understand, and synthesize all human-machine information. The last time a biological human was able to grasp all human scientific knowledge was hundreds of years ago.

Another advantage of machine intelligence is that it can consistently perform at peak levels and can combine peak skills. Among humans one person may have mastered music composition, while another may have mastered transistor design, but given the fixed architecture of our brains we do not have the capacity (or the time) to develop and utilize the highest level of skill in every increasingly specialized area. Humans also vary a great deal in a particular skill, so that when we speak, say, of human levels of composing music, do we mean Beethoven, or do we mean the average person? Nonbiological intelligence will be able to match and exceed peak human skills in each area.

For these reasons, once a computer is able to match the subtlety and range of human intelligence, it will necessarily soar past it, and then continue its double- exponential ascent.

A key question regarding the Singularity is whether the "chicken" (strong AI) or the "egg" (nanotechnology) will come first. In other words, will strong AI lead to full nanotechnology (molecular manufacturing assemblers that can turn information into physical products), or will full nanotechnology lead to strong AI? The logic of the first premise is that strong AI would imply superhuman AI for the reasons just cited; and superhuman AI would be in a position to solve any remaining design problems required to implement full nanotechnology.

The second premise is based on the realization that the hardware requirements for strong AI will be met by nanotechnology-based computation. Likewise the software requirements will be facilitated by nanobots that could create highly detailed scans of human brain functioning and thereby achieve the completion of reverse engineering the human brain.

Both premises are logical; it's clear that either technology can assist the other. The reality is that progress in both areas will necessarily use our most advanced tools, so advances in each field will simultaneously

facilitate the other. However, I do expect that full MNT will emerge prior to strong AI, but only by a few years (around 2025 for nanotechnology, around 2029 for strong AI).

As revolutionary as nanotechnology will be, strong AI will have far more profound consequences. Nanotechnology is powerful but not necessarily intelligent. We can devise ways of at least trying to manage the enormous powers of nanotechnology, but superintelligence innately cannot be controlled.

Runaway AI. Once strong AI is achieved, it can readily be advanced and its powers multiplied, as that is the fundamental nature of machine abilities. As one strong AI immediately begets many strong AIs, the latter access their own design, understand and improve it, and thereby very rapidly evolve into a yet more capable, more intelligent AI, with the cycle repeating itself indefinitely. Each cycle not only creates a more intelligent AI but takes less time than the cycle before it, as is the nature of technological evolution (or any evolutionary process). The premise is that once strong AI is achieved, it will immediately become a runaway phenomenon of rapidly escalating superintelligence.[4]

My own view is only slightly different. The logic of runaway AI is valid, but we still need to consider the timing. Achieving human levels in a machine will not *immediately* cause a runaway phenomenon. Consider that a human level of intelligence has limitations. We have examples of this today—about six billion of them. Consider a scenario in which you took one hundred humans from, say, a shopping mall. This group would constitute examples of reasonably well educated humans. Yet if this group was presented with the task of improving human intelligence, it wouldn't get very far, even if provided with the templates of human intelligence. It would probably have a hard time creating a simple computer. Speeding up the thinking and expanding the memory capacities of these one hundred humans would not immediately solve this problem.

I pointed out above that machines will match (and quickly exceed) peak human skills in each area of skill. So instead, let's take one hundred scientists and engineers. A group of technically trained people with the right backgrounds would be capable of improving accessible designs. If a machine attained equivalence to one hundred (and eventually one thousand, then one million) technically trained humans, each operating much faster than a biological human, a rapid acceleration of intelligence would ultimately follow.

However, this acceleration won't happen immediately when a computer passes the Turing test. The Turing test is comparable to matching the capabilities of an average, educated human and thus is closer to the example of humans from a shopping mall. It will take time for computers to master all of the requisite skills and to marry these skills with all the necessary knowledge bases.

Once we've succeeded in creating a machine that can pass the Turing test (around 2029), the succeeding period will be an era of consolidation in which nonbiological intelligence will make rapid gains. However, the extraordinary expansion contemplated for the Singularity, in which human intelligence is multiplied by billions, won't take place until the mid-2040s (as discussed in chapter 3 of *The Singularity is Near*).

The AI Winter

There's this stupid myth out there that A.I. has failed, but A.I. is everywhere around you every second of the day. People just don't notice it. You've got A.I. systems in cars, tuning the parameters of the fuel injection systems. When you land in an airplane, your gate gets chosen by an A.I. scheduling system. Every time you use a piece of Microsoft software, you've got an A.I. system trying to figure out what you're doing, like writing a letter, and it does a pretty damned good job. Every time you see a movie with computer-generated characters, they're all little A.I. characters behaving as a group. Every time you play a video game, you're playing against an A.I. system. —Rodney Brooks, director of the MIT AI Lab[5]

I still run into people who claim that artificial intelligence withered in the 1980s, an argument that is comparable to insisting that the Internet died in the dot-com bust of the early 2000s.[6] The bandwidth and price-performance of Internet technologies, the number of nodes (servers), and the dollar volume of e-commerce all accelerated smoothly through the boom as well as the bust and the period since. The same has been true for AI.

The technology hype cycle for a paradigm shift—railroads, AI, Internet, telecommunications, possibly now nanotechnology—typically starts with a period of unrealistic expectations based on a lack of understanding of all the enabling factors required. Although utilization of the new paradigm does increase exponentially, early growth is slow until the knee of the exponential-growth curve is realized. While the widespread expectations for revolutionary change are accurate, they are incorrectly timed. When the prospects do not quickly pan out, a period of disillusionment sets in. Nevertheless exponential growth continues unabated, and years later a more mature and more realistic transformation does occur.

We saw this in the railroad frenzy of the nineteenth century, which was followed by widespread bankruptcies. (I have some of these early unpaid railroad bonds in my collection of historical documents.) And we are still feeling the effects of the e-commerce and telecommunications busts of several years ago, which helped fuel a recession from which we are now recovering.

AI experienced a similar premature optimism in the wake of programs such as the 1957 General Problem Solver created by Allen Newell, J. C. Shaw, and Herbert Simon, which was able to find proofs for theorems that had stumped mathematicians such as Bertrand Russell, and early programs from the MIT Artificial Intelligence Laboratory, which could answer SAT questions (such as analogies and story problems) at the level of college students.[7] A rash of AI companies occurred in the 1970s, but when profits did not materialize there was an AI "bust" in the 1980s, which has become known as the "AI winter." Many observers still think that the AI winter was the end of the story and that nothing has since come of the AI field.

Yet today many thousands of AI applications are deeply embedded in the infrastructure of every industry. Most of these applications were research projects ten to fifteen years ago. People who ask, "Whatever happened to AI?" remind me of travelers to the rain forest who wonder, "Where are all these many species that are supposed to live here?" when hundreds of

species of flora and fauna are flourishing only a few dozen meters away, deeply integrated into the local ecology.

We are well into the era of "narrow AI," which refers to artificial intelligence that performs a useful and specific function that once required human intelligence to perform, and does so at human levels or better. Often narrow AI systems greatly exceed the speed of humans, as well as provide the ability to manage and consider thousands of variables simultaneously. I describe a broad variety of narrow AI examples below.

These time frames for AI's technology cycle (a couple of decades of growing enthusiasm, a decade of disillusionment, then a decade and a half of solid advance in adoption) may seem lengthy, compared to the relatively rapid phases of the Internet and telecommunications cycles (measured in years, not decades), but two factors must be considered. First, the Internet and telecommunications cycles were relatively recent, so they are more affected by the acceleration of paradigm shift (as discussed in chapter 1 of *The Singularity is Near*). So recent adoption cycles (boom, bust, and recovery) will be much faster than ones that started forty years ago. Second, the AI revolution is the most profound transformation that human civilization will experience, so it will take longer to mature than less complex technologies. It is characterized by the mastery of the most important and most powerful attribute of human civilization, indeed of the entire sweep of evolution on our planet: intelligence.

It's the nature of technology to understand a phenomenon and then engineer systems that concentrate and focus that phenomenon to greatly amplify it. For example, scientists discovered a subtle property of curved surfaces known as Bernoulli's principle: a gas (such as air) travels more quickly over a curved surface than over a flat surface. Thus, air pressure over a curved surface is lower than over a flat surface. By understanding, focusing, and amplifying the implications of this subtle observation, our engineering created all of aviation. Once we understand the principles of intelligence, we will have a similar opportunity to focus, concentrate, and amplify its powers.

As I reviewed in chapter 4 of *The Singularity is Near*, every aspect of understanding, modeling, and simulating the human brain is accelerating: the price-performance and temporal and spatial resolution of brain scanning, the amount of data and knowledge available about brain function, and the sophistication of the models and simulations of the brain's varied regions.

We already have a set of powerful tools that emerged from AI research and that have been refined and improved over several decades of development. The brain reverse-engineering project will greatly augment this toolkit by also providing a panoply of new, biologically inspired, self-organizing techniques. We will ultimately be able to apply engineering's ability to focus and amplify human intelligence vastly beyond the hundred trillion extremely slow interneuronal connections that each of us struggles with today. Intelligence will then be fully subject to the law of accelerating returns, which is currently doubling the power of information technologies every year.

An underlying problem with artificial intelligence that I have personally experienced in my forty years in this area is that as soon as an AI technique works, it's no longer considered AI and is spun off as its own

field (for example, character recognition, speech recognition, machine vision, robotics, data mining, medical informatics, automated investing).

Computer scientist Elaine Rich defines AI as "the study of how to make computers do things at which, at the moment, people are better." Rodney Brooks, director of the MIT AI Lab, puts it a different way: "Every time we figure out a piece of it, it stops being magical; we say, *Oh, that's just a computation*." I am also reminded of Watson's remark to Sherlock Holmes, "I thought at first that you had done something clever, but I see that there was nothing in it after all."[8] That has been our experience as AI scientists. The enchantment of intelligence seems to be reduced to "nothing" when we fully understand its methods. The mystery that is left is the intrigue inspired by the remaining, not as yet understood methods of intelligence.

AI's Toolkit

AI is the study of techniques for solving exponentially hard problems in polynomial time by exploiting knowledge about the problem domain.
—Elaine Rich

It has only been recently that we have been able to obtain sufficiently detailed models of how human brain regions function to influence AI design. Prior to that, in the absence of tools that could peer into the brain with sufficient resolution, AI scientists and engineers developed their own techniques. Just as aviation engineers did not model the ability to fly on the flight of birds, these early AI methods were not based on reverse engineering natural intelligence.

A small sample of these approaches is reviewed here. Since their adoption, they have grown in sophistication, which has enabled the creation of practical products that avoid the fragility and high error rates of earlier systems.

Expert systems. In the 1970s AI was often equated with one specific method: expert systems. This involves the development of specific logical rules to simulate the decision-making processes of human experts. A key part of the procedure entails knowledge engineers interviewing domain experts such as doctors and engineers to codify their decision-making rules.

There were early successes in this area, such as medical diagnostic systems that compared well to human physicians, at least in limited tests. For example, a system called MYCIN, which was designed to diagnose and recommend remedial treatment for infectious diseases, was developed through the 1970s. In 1979 a team of expert evaluators compared diagnosis and treatment recommendations by MYCIN to those of human doctors and found that MYCIN did as well as or better than any of the physicians.[9]

It became apparent from this research that human decision making typically is based not on definitive logic rules but rather on "softer" types of evidence. A dark spot on a medical imaging test may suggest cancer, but other factors such as its exact shape, location, and contrast are likely to influence a diagnosis. The hunches of human decision making are usually influenced by combining many pieces of evidence from prior experience, none definitive by itself. Often we are not even consciously aware of many of the rules that we use.

By the late 1980s expert systems were incorporating the idea of uncertainty and could combine many sources of probabilistic evidence to make a decision. The MYCIN system pioneered this approach. A typical MYCIN "rule" reads:

If the infection which requires therapy is meningitis, and the type of the infection is fungal, and organisms were not seen on the stain of the culture, and the patient is not a compromised host, and the patient has been to an area that is endemic for coccidiomycoses, and the race of the patient is Black, Asian, or Indian, and the cryptococcal antigen in the csf test was not positive, THEN there is a 50 percent chance that cryptococcus is not one of the organisms which is causing the infection.

Although a single probabilistic rule such as this would not be sufficient by itself to make a useful statement, by combining thousands of such rules the evidence can be marshaled and combined to make reliable decisions.

Probably the longest-running expert system project is CYC (for enCYClopedic), created by Doug Lenat and his colleagues at Cycorp. Initiated in 1984, CYC has been coding commonsense knowledge to provide machines with an ability to understand the unspoken assumptions underlying human ideas and reasoning. The project has evolved from hard-coded logical rules to probabilistic ones and now includes means of extracting knowledge from written sources (with human supervision). The original goal was to generate one million rules, which reflects only a small portion of what the average human knows about the world. Lenat's latest goal is for CYC to master "100 million things, about the number a typical person knows about the world by 1997."[10]

Another ambitious expert system is being pursued by Darryl Macer, associate professor of Biological Sciences at the University of Tsukuba in Japan. He plans to develop a system incorporating all human ideas.[11] One application would be to inform policy makers of which ideas are held by which community.

Bayesian nets. Over the last decade a technique called Bayesian logic has created a robust mathematical foundation for combining thousands or even millions of such probabilistic rules in what are called "belief networks" or Bayesian nets. Originally devised by English mathematician Thomas Bayes, and published posthumously in 1763, the approach is intended to determine the likelihood of future events based similar occurrences in the past.[12] Many expert systems based on Bayesian techniques gather data from experience in an ongoing fashion, thereby continually learning and improving their decision making.

The most promising type of spam filters are based on this method. I personally use a spam filter called SpamBayes, which trains itself on e-mail that you have identified as either "spam" or "okay."[13] You start out by presenting a folder of each to the filter. It trains its Bayesian belief network on these two files and analyzes the patterns of each, thus enabling it to automatically move subsequent e-mail into the proper category. It continues to train itself on every subsequent e-mail, especially when it's corrected by the user. This filter has made the spam situation manageable for me, which is saying a lot, as it weeds out two hundred to three hundred spam messages each day, letting over one hundred "good" messages through. Only about 1 percent of the messages

it identifies as "okay" are actually spam; it almost never marks a good message as spam. The system is almost as accurate as I would be and much faster.

Markov models. Another method that is good at applying probabilistic networks to complex sequences of information involves Markov models.[14] Andrei Andreyevich Markov (1856–1922), a renowned mathematician, established a theory of "Markov chains," which was refined by Norbert Wiener (1894–1964) in 1923. The theory provided a method to evaluate the likelihood that a certain sequence of events would occur. It has been popular, for example, in speech recognition, in which the sequential events are phonemes (parts of speech). The Markov models used in speech recognition code the likelihood that specific patterns of sound are found in each phoneme, how the phonemes influence each other, and likely orders of phonemes. The system can also include probability networks on higher levels of language, such as the order of words. The actual probabilities in the models are trained on actual speech and language data, so the method is self-organizing.

Markov modeling was one of the methods my colleagues and I used in our own speech-recognition development.[15] Unlike phonetic approaches, in which specific rules about phoneme sequences are explicitly coded by human linguists, we did not tell the system that there are approximately forty-four phonemes in English, nor did we tell it what sequences of phonemes were more likely than others. We let the system discover these "rules" for itself from thousands of hours of transcribed human speech data. The advantage of this approach over hand-coded rules is that the models develop subtle probabilistic rules of which human experts are not necessarily aware.

Neural nets. Another popular self-organizing method that has also been used in speech recognition and a wide variety of other pattern-recognition tasks is neural nets. This technique involves simulating a simplified model of neurons and interneuronal connections. One basic approach to neural nets can be described as follows. Each point of a given input (for speech, each point represents two dimensions, one being frequency and the other time; for images, each point would be a pixel in a two-dimensional image) is randomly connected to the inputs of the first layer of simulated neurons. Every connection has an associated synaptic strength, which represents its importance and which is set at a random value. Each neuron adds up the signals coming into it. If the combined signal exceeds a particular threshold, the neuron fires and sends a signal to its output connection; if the combined input signal does not exceed the threshold, the neuron does not fire, and its output is zero. The output of each neuron is randomly connected to the inputs of the neurons in the next layer. There are multiple layers (generally three or more), and the layers may be organized in a variety of configurations. For example, one layer may feed back to an earlier layer. At the top layer, the output of one or more neurons, also randomly selected, provides the answer (For an algorithmic description of neural nets, see this endnote.[16]).

Since the neural-net wiring and synaptic weights are initially set randomly, the answers of an untrained neural net will be random. The key to a neural net, therefore, is that it must learn its subject matter. Like the mammalian brains on which it's loosely modeled, a neural net starts out ignorant. The neural net's teacher—which may be a human, a computer program, or perhaps another, more mature neural net that has

already learned its lessons—rewards the student neural net when it generates the right output and punishes it when it does not. This feedback is in turn used by the student neural net to adjust the strengths of each interneuronal connection. Connections that were consistent with the right answer are made stronger. Those that advocated a wrong answer are weakened. Over time, the neural net organizes itself to provide the right answers without coaching. Experiments have shown that neural nets can learn their subject matter even with unreliable teachers. If the teacher is correct only 60 percent of the time, the student neural net will still learn its lessons.

A powerful, well-taught neural net can emulate a wide range of human pattern-recognition faculties. Systems using multilayer neural nets have shown impressive results in a wide variety of pattern-recognition tasks, including recognizing handwriting, human faces, fraud in commercial transactions such as credit-card charges, and many others. In my own experience in using neural nets in such contexts, the most challenging engineering task is not coding the nets but in providing automated lessons for them to learn their subject matter.

The current trend in neural nets is to take advantage of more realistic and more complex models of how actual biological neural nets work, now that we are developing detailed models of neural functioning from brain reverse engineering.[17] Since we do have several decades of experience in using self-organizing paradigms, new insights from brain studies can be quickly adapted to neural-net experiments.

Neural nets are also naturally amenable to parallel processing, since that is how the brain works. The human brain does not have a central processor that simulates each neuron. Rather, we can consider each neuron and each interneuronal connection to be an individual slow processor. Extensive work is under way to develop specialized chips that implement neural-net architectures in parallel to provide substantially greater throughput.[18]

Genetic algorithms (GAs). Another self-organizing paradigm inspired by nature is genetic, or evolutionary, algorithms, which emulate evolution, including sexual reproduction and mutations. Here is a simplified description of how they work. First, determine a way to code possible solutions to a given problem. If the problem is optimizing the design parameters for a jet engine, define a list of the parameters (with a specific number of bits assigned to each parameter). This list is regarded as the genetic code in the genetic algorithm. Then randomly generate thousands or more genetic codes. Each such genetic code (which represents one set of design parameters) is considered a simulated "solution" organism.

Now evaluate each simulated organism in a simulated environment by using a defined method to evaluate each set of parameters. This evaluation is a key to the success of a genetic algorithm. In our example, we would apply each solution organism to a jet-engine simulation and determine how successful that set of parameters is, according to whatever criteria we are interested in (fuel consumption, speed, and so on). The best solution organisms (the best designs) are allowed to survive, and the rest are eliminated.

Now have each of the survivors multiply themselves until they reach the same number of solution creatures. This is done by simulating sexual

reproduction. In other words, each new offspring solution draws part of its genetic code from one parent and another part from a second parent. Usually no distinction is made between male or female organisms; it's sufficient to generate an offspring from two arbitrary parents. As they multiply, allow some mutation (random change) in the chromosomes to occur.

We've now defined one generation of simulated evolution; now repeat these steps for each subsequent generation. At the end of each generation determine how much the designs have improved. When the improvement in the evaluation of the design creatures from one generation to the next becomes very small, we stop this iterative cycle of improvement and use the best design(s) in the last generation .(For an algorithmic description of genetic algorithms, see this endnote[19]).

The key to a GA is that the human designers don't directly program a solution; rather, they let one emerge through an iterative process of simulated competition and improvement. As we discussed, biological evolution is smart but slow, so to enhance its intelligence we retain its discernment while greatly speeding up its ponderous pace. The computer is fast enough to simulate many generations in a matter of hours or days or weeks. But we have to go through this iterative process only once; once we have let this simulated evolution run its course, we can apply the evolved and highly refined rules to real problems in a rapid fashion.

Like neural nets GAs are a way to harness the subtle but profound patterns that exist in chaotic data. A key requirement for their success is a valid way of evaluating each possible solution. This evaluation needs to be fast because it must take account of many thousands of possible solutions for each generation of simulated evolution.

GAs are adept at handling problems with too many variables to compute precise analytic solutions. The design of a jet engine, for example, involves more than one hundred variables and requires satisfying dozens of constraints. GAs used by researchers at General Electric were able to come up with engine designs that met the constraints more precisely than conventional methods.

When using GAs you must, however, be careful what you ask for. University of Sussex researcher Jon Bird used a GA to optimally design an oscillator circuit. Several attempts generated conventional designs using a small number of transistors, but the winning design was not an oscillator at all but a simple radio circuit. Apparently the GA discovered that the radio circuit picked up an oscillating hum from a nearby computer.[20] The GA's solution worked only in the exact location on the table where it was asked to solve the problem.

Genetic algorithms, part of the field of chaos or complexity theory, are being increasingly used to solve otherwise intractable business problems, such as optimizing complex supply chains. This approach is beginning to supplant more analytic methods throughout industry. (See examples below.) The paradigm is also adept at recognizing patterns, and is often combined with neural nets and other self-organizing methods. It's also a reasonable way to write computer software, particularly software that needs to find delicate balances for competing resources.

In the novel *usr/bin/god*, Cory Doctorow, a leading science-fiction writer, uses an intriguing variation of a GA to evolve an AI. The GA generates a

large number of intelligent systems based on various intricate combinations of techniques, with each combination characterized by its genetic code. These systems then evolve using a GA.

The evaluation function works as follows: each system logs on to various human chat rooms and tries to pass for a human, basically a covert Turing test. If one of the humans in a chat room says something like "What are you, a chatterbot?" (chatterbot meaning an automatic program, that at today's level of development is expected to not understand language at a human level), the evaluation is over, that system ends its interactions, and reports its score to the GA. The score is determined by how long it was able to pass for human without being challenged in this way. The GA evolves more and more intricate combinations of techniques that are increasingly capable of passing for human.

The main difficulty with this idea is that the evaluation function is fairly slow, although it will take an appreciable amount of time only once the systems are reasonably intelligent. Also, the evaluations can take place largely in parallel. It's an interesting idea and may actually be a useful method to finish the job of passing the Turing test, once we get to the point where we have sufficiently sophisticated algorithms to feed into such a GA, so that evolving a Turing-capable AI is feasible.

Recursive search. Often we need to search through vast number of combinations of possible solutions to solve a given problem. A classic example is in playing games such as chess. As a player considers her next move, she can list all of her possible moves, and then, for each such move, all possible countermoves by the opponent, and so on. It is difficult, however, for human players to keep a huge "tree" of move-countermove sequences in their heads, and so they rely on pattern recognition—recognizing situations based on prior experience—whereas machines use logical analysis of millions of moves and countermoves.

Such a logical tree is at the heart of most game-playing programs. Consider how this is done. We construct a program called Pick Best Next Step to select each move. Pick Best Next Step starts by listing all of the possible moves from the current state of the board. (If the problem was solving a mathematical theorem, rather than game moves, the program would list all of the possible next steps in a proof.) For each move the program constructs a hypothetical board that reflects what would happen if we made this move. For each such hypothetical board, we now need to consider what our opponent would do if we made this move. Now recursion comes in, because Pick Best Next Step simply calls Pick Best Next Step (in other words, itself) to pick the best move for our opponent. In calling itself, Pick Best Next Step then lists all of the legal moves for our opponent.

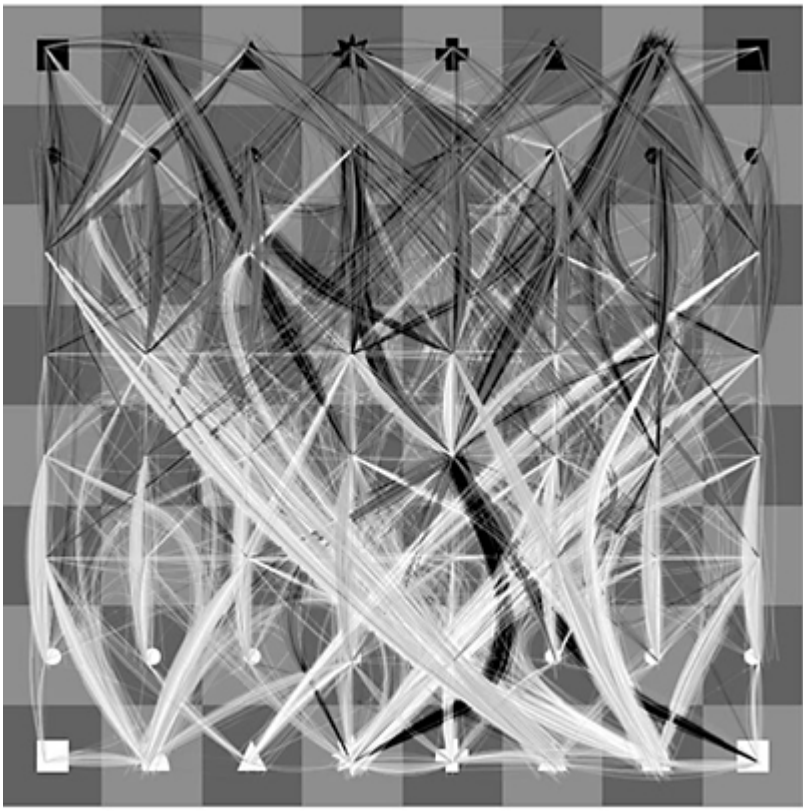
The program keeps calling itself, looking ahead as many moves as we have time to consider, which results in the generation of a huge move-countermove tree. This is another example of exponential growth, because to look ahead an additional move (or countermove) requires multiplying the amount of available computation by about five. Key to the success of the recursive formula is pruning this huge tree of possibilities and ultimately stopping its growth. In the game context, if a board looks hopeless for either side, the program can stop the expansion of the move-countermove tree from that point (called a "terminal leaf" of the tree) and consider the most recently considered move to be a likely win

or loss. When all of these nested program calls are completed, the program will have determined the best possible move for the current actual board within the limits of the depth of recursive expansion that it had time to pursue, and the quality of its pruning algorithm (For an algorithmic description of recursive search, see this endnote.[21]).

The recursive formula is often effective at mathematics. Rather than game moves, the "moves" are the axioms of the field of math being addressed, as well as previously proved theorems. The expansion at each point is the possible axioms (or previously proved theorems) that can be applied to a proof at each step. (This was the approach used by Newell, Shaw, and Simons's General Problem Solver.)

From these examples it may appear that recursion is well suited only for problems in which we have crisply defined rules and objectives. But it has also shown promise in computer generation of artistic creations. For example, a program I designed called Ray Kurzweil's Cybernetic Poet uses a recursive approach.[22] The program establishes a set of goals for each word—achieving a certain rhythmic pattern, poem structure, and word choice that is desirable at that point in the poem. If the program is unable to find a word that meets these criteria, it backs up and erases the previous word it has written, reestablishes the criteria it had originally set for the word just erased, and goes from there. If that also leads to a dead end, it backs up again, thus moving backwards and forwards. Eventually, it forces itself to make up its mind by relaxing some of the constraints if all paths lead to dead ends.

Black (computer)...is deciding on a move



White (you)

Deep Fritz Draws: Are Humans Getting

Smarter, or Are Computers Getting Stupider?

We find one example of qualitative improvements in software in the world of computer chess, which, according to common wisdom, is governed only by the brute-force expansion of computer hardware. In a chess tournament in October 2002 with top-ranking human player Vladimir Kramnik, the Deep Fritz software achieved a draw. I point out that Deep Fritz has available only about 1.3 percent of the brute-force computation as the previous computer champion, Deep Blue. Despite that, it plays chess at about the same level because of its superior pattern recognition–based pruning algorithm (see below). In six years a program like Deep Fritz will again achieve Deep Blue's ability to analyze two hundred million board positions per second.

Deep Fritz–like chess programs running on ordinary personal computers will routinely defeat all humans later in this decade.

In *The Age of Intelligent Machines*, which I wrote between 1986 and 1989, I predicted that a computer would defeat the human world chess champion by the end of the 1990s. I also noted that computers were gaining about forty-five points per year in their chess ratings, whereas the best human playing was essentially fixed, so this projected a crossover point in 1998. Indeed, Deep Blue did defeat Gary Kasparov in a highly publicized tournament in 1997.

Yet in the Deep Fritz–Kramnik match, the current reigning computer program was able to achieve only a tie. Five years had passed since Deep Blue's victory, so what are we to make of this situation? Should we conclude that:

1. Humans are getting smarter, or at least better at chess?

2. Computers are getting worse at chess? If so, should we conclude that the much- publicized improvement in computer speed over the past five years was not all it was cracked up to be? Or, that computer software is getting worse, at least in chess?

The specialized-hardware advantage. Neither of the above conclusions is warranted. The correct conclusion is that software is getting better because Deep Fritz essentially matched the performance of Deep Blue, yet with far smaller computational resources. To gain some insight into these questions, we need to examine a few essentials. When I wrote my predictions of computer chess in the late 1980s, Carnegie Mellon University was embarked on a program to develop specialized chips for conducting

the "minimax" algorithm (the standard game-playing method that relies on building trees of move-countermove sequences, and then evaluating the terminal-leaf positions of each branch of the tree) specifically for chess moves.

Based on this specialized hardware CMU's 1988 chess machine, HiTech, was able to analyze 175,000 board positions per second. It achieved a chess rating of 2,359, only about 440 points below the human world champion.

A year later, in 1989, CMU's Deep Thought machine increased this capacity to one million board positions per second and achieved a rating of 2,400. IBM eventually took over the project and renamed it Deep Blue but kept the basic CMU architecture. The version of Deep Blue that defeated Kasparov in 1997 had 256 special-purpose chess processors working in parallel, which analyzed two hundred million board positions per second.

It is important to note the use of specialized hardware to accelerate the specific calculations needed to generate the minimax algorithm for chess moves. It's well-known to computer-systems designers that specialized hardware generally can implement a specific algorithm at least one hundred times faster than a general-purpose computer. Specialized ASICs (Application-specific Integrated Circuits) require significant development efforts and costs, but for critical calculations that are needed on a repetitive basis (for example, decoding MP3 files or rendering graphics primitives for video games), this expenditure can be well worth the investment.

Deep Blue versus Deep Fritz. Because there had always been a great deal of focus on the milestone of a computer's being able to defeat a human opponent, support was available for investing in special-purpose chess circuits. Although there was some lingering controversy regarding the parameters of the Deep Blue–Kasparov match, the level of interest in computer chess waned considerably after 1997. After all, the goal had been achieved, and there was little point in beating a dead horse. IBM canceled work on the project, and there has been no work on specialized chess chips since that time. The focus of research in the various domains spun out of AI has been placed instead on problems of greater consequence, such as guiding airplanes, missiles, and factory robots, understanding natural language, diagnosing electrocardiograms and blood-cell images, detecting credit-card fraud, and a myriad of other successful narrow AI applications.

Computer hardware has nonetheless continued its exponential increase, with personal-computer speeds

doubling every year since 1997. Thus the general-purpose Pentium processors used by Deep Fritz are about thirty-two times faster than processors in 1997. Deep Fritz uses a network of only eight personal computers, so the hardware is equivalent to 256 1997-class personal computers. Compare that to Deep Blue, which used 256 specialized chess processors, each of which was about one hundred times faster than 1997 personal computers (of course, only for computing chess minimax). So Deep Blue was 25,600 times faster than a 1997 PC and one hundred times faster than Deep Fritz. This analysis is confirmed by the reported speeds of the two systems: Deep Blue can analyze 200 million board positions per second compared to only about 2.5 million for Deep Fritz.

Significant software gains. So what can we say about the software in Deep Fritz? Although chess machines are usually referred to as examples of brute-force calculation, there is one important aspect of these systems that does require qualitative judgment. The combinatorial explosion of possible move-countermove sequences is rather formidable.

In *The Age of Intelligent Machines* I estimated that it would take about forty billion years to make one move if we failed to prune the move-countermove tree and attempted to make a "perfect" move in a typical game. (Assuming about thirty moves each in a typical game and about eight possible moves per play, we have 8^{30} possible move sequences; analyzing one billion move sequences per second would take 10^{18} seconds or forty billion years.) Thus a practical system needs to continually prune away unpromising lines of action. This requires insight and is essentially a pattern-recognition judgment.

Humans, even world-class chess masters, perform the minimax algorithm extremely slowly, generally performing less than one move-countermove analysis per second. So how is it that a chess master can compete at all with computer systems? The answer is that we possess formidable powers of pattern recognition, which enable us to prune the tree with great insight.

It's precisely in this area that Deep Fritz has improved considerably over Deep Blue. Deep Fritz has only slightly more computation available than CMU's Deep Thought yet is rated almost 400 points higher.

Are human chess players doomed? Another prediction I made in *The Age of Intelligent Machines* was that once computers did perform as well or better as humans in chess, we would either think more of computer intelligence, less of human intelligence, or less of chess, and that if history is a guide, the last of

these would be the likely outcome. Indeed, that is precisely what happened. Soon after Deep Blue's victory we began to hear a lot about how chess is really just a simple game of calculating combinations and that the computer victory just demonstrated that it was a better calculator.

The reality is slightly more complex. The ability of humans to perform well in chess is clearly not due to our calculating prowess, at which we are in fact rather poor. We use instead a quintessentially human form of judgment. For this type of qualitative judgment, Deep Fritz represents genuine progress over earlier systems. (Incidentally, humans have made no progress in the last five years, with the top human scores remaining just below 2,800. Kasparov is rated at 2,795 and Kramnik at 2,794.)

Where we go from here? Now that computer chess is relying on software running on ordinary personal computers, chess programs will continue to benefit from the ongoing acceleration of computer power. By 2009 a program like Deep Fritz will again achieve Deep Blue's ability to analyze two hundred million board positions per second. With the opportunity to harvest computation on the Internet, we will be able to achieve this potential several years sooner than 2009. (Internet harvesting of computers will require more ubiquitous broadband communication, but that's coming, too).

With these inevitable speed increases, as well as ongoing improvements in pattern recognition, computer chess ratings will continue to edge higher. Deep Fritz–like programs running on ordinary personal computers will routinely defeat all humans later in this decade. Then we'll really lose interest in chess.

Combining methods. The most powerful approach to building robust AI systems is to combine approaches, which is how the human brain works. As we discussed, the brain is not one big neural net but instead consists of hundreds of regions, each of which is optimized for processing information in a different way. None of these regions by itself operates at what we would consider human levels of performance, but clearly by definition the overall system does exactly that.

I've used this approach in my own AI work, especially in pattern recognition. In speech recognition, for example, we implemented a number of different pattern-recognition systems based on different paradigms. Some were specifically programmed with knowledge of phonetic and linguistic constraints from experts. Some were based on rules to parse sentences (which involves creating sentence diagrams showing word usage, similar to the diagrams taught in grade school). Some were based on self-organizing techniques, such as Markov models, trained on extensive libraries of recorded and annotated human speech. We then programmed a software "expert manager" to learn the strengths

and weaknesses of the different "experts" (recognizers) and combine their results in optimal ways. In this fashion, a particular technique that by itself might produce unreliable results can nonetheless contribute to increasing the overall accuracy of the system.

There are many intricate ways to combine the varied methods in AI's toolbox. For example, one can use a genetic algorithm to evolve the optimal topology (organization of nodes and connections) for a neural net or a Markov model. The final output of the GA-evolved neural net can then be used to control the parameters of a recursive search algorithm. We can add in powerful signal- and image-processing techniques that have been developed for pattern-processing systems. Each specific application calls for a different architecture. Computer science professor and AI entrepreneur Ben Goertzel has written a series of books and articles that describe strategies and architectures for combining the diverse methods underlying intelligence. His "Novamente" architecture is intended to provide a framework for general purpose AI.[23]

The above basic descriptions provide only a glimpse into how increasingly sophisticated current AI systems are designed. It's beyond the scope of this book to provide a comprehensive description of the techniques of AI, and even a doctoral program in computer science is unable to cover all of the varied approaches in use today.

Many of the examples of real-world narrow AI systems described in the next section use a variety of methods integrated and optimized for each particular task. Narrow AI is strengthening as a result of several concurrent trends: continued exponential gains in computational resources, extensive real-world experience with thousands of applications, and fresh insights into how the human brain makes intelligent decisions.

A Narrow AI Sampler

When I wrote my first AI book, *The Age of Intelligent Machines* in the late 1980s, I had to conduct extensive investigations to find a few successful examples of AI in practice. The Internet was not yet prevalent, so I had to go to real libraries and visit the AI research centers in the United States, Europe, and Asia. I included in my book pretty much all of the reasonable examples I could identify. In my research for this book my experience has been altogether different. I have been inundated with thousands of compelling examples. In our reporting on the KurzweilAI.net Web site, we feature several dramatic systems every day.[24]

A 2003 study by Business Communication Company projected a $21 billion market by 2007 for AI applications, with average annual growth of 12.2 percent from 2002 to 2007.[25] Leading industries for AI applications include business intelligence, customer relations, finance, defense and domestic security, and education. Here is a small sample of narrow AI in action.

Military and intelligence. The U.S. military has been an avid user of AI systems. Pattern-recognition software systems guide autonomous weapons such as cruise missiles, which can fly thousands of miles to find a specific building or even a specific window.[26] Although the relevant details of the terrain that the missile flies over are programmed ahead of time, variations in weather, ground cover, and other factors require a flexible level of real-time image recognition.

The army has developed prototypes of self-organizing communication networks (called "mesh networks") to automatically configure many thousands of communication nodes when a platoon is dropped into a new location.[27]

Expert systems incorporating Bayesian networks and GAs are used to optimize complex supply chains that coordinate millions of provisions, supplies, and weapons based on rapidly changing battlefield requirements.

AI systems are routinely employed to simulate the performance of weapons, including nuclear bombs and missiles.

Advance warning of the September 11, 2001, terrorist attacks was apparently detected by the National Security Agency's AI-based Echelon system, which analyzes the agency's extensive monitoring of communications traffic.[28] Unfortunately, Echelon's warnings were not reviewed by human agents until it was too late.

The 2002 military campaign in Afghanistan saw the debut of the armed Predator, an unmanned robotic flying fighter. Although the Air Force's Predator had been under development for many years, arming it with Army-supplied missiles was a last-minute improvisation that proved remarkably successful. In the Iraq war that began in 2003 the armed Predator (operated by the CIA) and other flying unmanned aerial vehicles (UAVs) destroyed thousands of enemy tanks and missile sites.

All of the military services are using robots. The army utilizes them to search caves (in Afghanistan) and buildings. The navy uses small robotic ships to protect its aircraft carriers. As I discuss in chapter 6 of *The Singularity is Near*, moving soldiers away from battle is a rapidly growing trend.

Space exploration. NASA is building self-understanding into the software controlling its unmanned spacecraft. Because Mars is about three light-minutes from Earth, and Jupiter around forty light-minutes (depending on the exact position of the planets), communication between spacecraft headed there and earthbound controllers is significantly delayed. For this reason it's important that the software controlling these missions have the capability of performing its own tactical decision making. To accomplish this NASA software is being designed to include a model of the software's own capabilities and those of the spacecraft, as well as the challenges each mission is likely to encounter. Such AI-based systems are capable of reasoning through new situations rather than just following preprogrammed rules. This approach enabled the craft Deep Space One in 1999 to use its own technical knowledge to devise a series of original plans to overcome a stuck switch that threatened to destroy its mission of exploring an asteroid.[29] The AI system's first plan failed to work, but its second plan saved the mission. "These systems have a commonsense model of the physics of their internal components," explains Brian Williams, coinventor of Deep Space One's autonomous software and now a scientist at MIT's Space Systems and AI laboratories. "[The spacecraft] can reason from that model to determine what is wrong and to know how to act."

Using a network of computers NASA used GAs to evolve an antenna design for three Space Technology 5 satellites that will study the Earth's

magnetic field. Millions of possible designs competed in the simulated evolution. According to NASA scientist and project leader Jason Lohn, "We are now using the [GA] software to design tiny microscopic machines, including gyroscopes, for spaceflight navigation. The software also may invent designs that no human designer would ever think of."[30]

Another NASA AI system learned on its own to distinguish stars from galaxies in very faint images with an accuracy surpassing that of human astronomers.

New land-based robotic telescopes are able to make their own decisions on where to look and how to optimize the likelihood of finding desired phenomena. Called "autonomous, semi-intelligent observatories," the systems can adjust to the weather, notice items of interest, and decide on their own to track them. They are able to detect very subtle phenomena, such as a star blinking for a nanosecond, which may indicate a small asteroid in the outer regions of our solar system passing in front of the light from that star.[31] One such system, called Moving Object and Transient Event Search System (MOTESS) has identified on its own 180 new asteroids and several comets during its first two years of operation. "We have an intelligent observing system," explained University of Exeter astronomer Alasdair Allan. "It thinks and reacts for itself, deciding whether something it has discovered is interesting enough to need more observations. If more observations are needed, it just goes ahead and gets them."

Similar systems are used by the military to automatically analyze data from spy satellites. Current satellite technology is capable of observing ground-level features about an inch in size and is not affected by bad weather, clouds, or darkness.[32] The massive amount of data continually generated would not be manageable without automated image recognition programmed to look for relevant developments.

Medicine. If you obtain an electrocardiogram (ECG) your doctor is likely to receive an automated diagnosis using pattern recognition applied to ECG recordings. My own company (Kurzweil Technologies) is working with United Therapeutics to develop a new generation of automated ECG analysis for long-term unobtrusive monitoring (via sensors embedded in clothing and wireless communication using a cell phone) of the early warning signs of heart disease.[33] Other pattern-recognition systems are used to diagnose a variety of imaging data.

Every major drug developer is using AI programs to do pattern recognition and intelligent data mining in the development of new drug therapies. For example SRI International is building flexible knowledge bases that encode everything we know about a dozen disease agents, including tuberculosis and *H. pylori* (the bacteria that cause ulcers).[34] The goal is to apply intelligent data-mining tools (software that can search for new relationships in data) to find new ways to kill or disrupt the metabolisms of these pathogens.

Similar systems are being applied to performing the automatic discovery of new therapies for other diseases, as well as understanding the function of genes and their roles in disease.[35] For example Abbott Laboratories claims that six human researchers in one of its new labs equipped with AI-based robotic and data-analysis systems are able to match the results of two hundred scientists in its older drug-development labs.[36]

Men with elevated prostate-specific antigen (PSA) levels typically undergo surgical biopsy, but about 75 percent of these men do not have prostate cancer. A new test, based on pattern recognition of proteins in the blood, would reduce this false positive rate to about 29 percent.[37] The test is based on an AI program designed by Correlogic Systems in Bethesda, Maryland, and the accuracy is expected to improve further with continued development.

Pattern recognition applied to protein patterns has also been used in the detection of ovarian cancer. The best contemporary test for ovarian cancer, called CA-125, employed in combination with ultrasound, misses almost all early-stage tumors. "By the time it is now diagnosed, ovarian cancer is too often deadly," says Emanuel Petricoin III, codirector of the Clinical Proteomics Program run by the FDA and the National Cancer Institute. Petricoin is the lead developer of a new AI-based test looking for unique patterns of proteins found only in the presence of cancer. In an evaluation involving hundreds of blood samples, the test was, according to Petricoin, "an astonishing 100% accurate in detecting cancer, even at the earliest stages."[38]

About 10 percent of all Pap-smear slides in the United States are analyzed by a self-learning AI program called FocalPoint, developed by TriPath Imaging. The developers started out by interviewing pathologists on the criteria they use. The AI system then continued to learn by watching expert pathologists. Only the best human diagnosticians were allowed to be observed by the program. "That's the advantage of an expert system," explains Bob Schmidt, TriPath's technical product manager. "It allows you to replicate your very best people."

Ohio State University Health System has developed a computerized physician order-entry (CPOE) system based on an expert system with extensive knowledge across multiple specialties.[39] The system automatically checks every order for possible allergies in the patient, drug interactions, duplications, drug restrictions, dosing guidelines, and appropriateness given information about the patient from the hospital's laboratory and radiology departments.

Science and math. A "robot scientist" has been developed at the University of Wales that combines an AI-based system capable of formulating original theories, a robotic system that can automatically carry out experiments, and a reasoning engine to evaluate results. The researchers provided their creation with a model of gene expression in yeast. The system "automatically originates hypotheses to explain observations, devises experiments to test these hypotheses, physically runs the experiments using a laboratory robot, interprets the results to falsify hypotheses inconsistent with the data, and then repeats the cycle."[40] The system is capable of improving its performance by learning from its own experience. The experiments designed by the robot scientist were three times less expensive than those designed by human scientists. A test of the machine against a group of human scientists showed that the discoveries made by the machine were comparable to those made by the humans.

Mike Young, director of biology at the University of Wales, was one of the human scientists who lost to the machine. He explains that "the robot did beat me, but only because I hit the wrong key at one point."

A long-standing conjecture in algebra was finally proved by an AI system at Argonne National Laboratory. Human mathematicians called the proof "creative."

Business, finance, and manufacturing. Companies in every industry are using AI systems to control and optimize logistics, detect fraud and money laundering, and perform intelligent data mining on the horde of information they gather each day. Wal-Mart, for example, gathers vast amounts of information from its transactions with shoppers. AI-based tools using neural nets and expert systems review this data to provide market-research reports for managers. This intelligent data mining allows them to make remarkably accurate predictions of the inventory required for each product in each store for each day.[41]

AI-based programs are routinely used to detect fraud in financial transactions. Future Route, an English company, for example, offers iHex, based on AI routines developed at Oxford University, to detect fraud in credit-card transactions and loan applications.[42] The system continuously generates and updates its own rules based on its experience. First Union Home Equity Bank in Charlotte, North Carolina, uses Loan Arranger, a similar AI-based system to decide whether to approve mortgage applications.[43]

The NASDAQ similarly uses a learning program called the Securities Observation, News Analysis, and Regulation (SONAR) system to monitor all trades for fraud as well as the possibility of insider trading.[44] As of the end of 2003 more than 180 incidents had been detected by SONAR and referred to the U.S. Securities and Exchange Commission and Department of Justice. These included several cases that later received significant news coverage.

Ascent Technology, founded by Patrick Winston, who directed MIT's AI Lab from 1972 through 1997, has designed an GA-based system called SmartAirport Operations Center (SAOC) that can optimize the complex logistics of an airport, such as balancing work assignments of hundreds of employees, making gate and equipment assignments, and managing a myriad of other details.[45] Winston points out that "figuring out ways to optimize a complicated situation is what genetic algorithms do." SAOC has raised productivity by approximately 30 percent in the airports where it has been implemented.

Ascent's first contract was to apply its AI techniques to managing the logistics for the 1991 Desert Storm campaign in Iraq. DARPA claimed that AI-based logistic-planning systems, including the Ascent system, resulted in more savings than the entire government research investment in AI over several decades.[46]

A recent trend in software is for AI systems to monitor a complex software system's performance, recognize malfunctions, and determine the best way to recover automatically without necessarily informing the human user. The idea stems from the realization that as software systems become more complex, like humans, they will never be perfect, and that eliminating all bugs is impossible. As humans, we use the same strategy: we don't expect to be perfect, but we usually try to recover from inevitable mistakes. "We want to stand this notion of systems management on its head," says Armando Fox, the head of Stanford

University's Software Infrastructures Group, who is working on what is now called "autonomic computing." Fox adds, "The system has to be able to set itself up, it has to optimize itself. It has to repair itself, and if something goes wrong, it has to know how to respond to external threats." IBM, Microsoft, and other software vendors are all developing systems that incorporate autonomic capabilities.

Manufacturing and robotics. Computer-integrated manufacturing (CIM) increasingly employs AI techniques to optimize the use of resources, streamline logistics, and reduce inventories through just-in-time purchasing of parts and supplies. A new trend in CIM systems is to use "case-based reasoning" rather than hard-coded, rule-based expert systems. Such reasoning codes knowledge as "cases," which are examples of problems with solutions. Initial cases are usually designed by the engineers, but the key to a successful case-based reasoning system is its ability to gather new cases from actual experience. The system is then able to apply the reasoning from its stored cases to new situations.

Robots are extensively used in manufacturing. The latest generation of robots uses flexible AI-based machine-vision systems—from companies such as Cognex Corporation in Natick, Massachusetts—that can respond flexibly to varying conditions. This reduces the need for precise setup for the robot to operate correctly. Brian Carlisle, CEO of Adept Technologies, a Livermore, California, factory-automation company, points out that "even if labor costs were eliminated [as a consideration], a strong case can still be made for automating with robots and other flexible automation. In addition to quality and throughput, users gain by enabling rapid product changeover and evolution that can't be matched with hard tooling."

One of AI's leading roboticists, Hans Moravec, has founded a company called Seegrid to apply his machine-vision technology to applications in manufacturing, materials handling, and military missions.[47] Moravec's software enables a device (a robot or just a material-handling cart) to walk or roll through an unstructured environment and in a single pass build a reliable "voxel" (three-dimensional pixel) map of the environment. The robot can then use the map and its own reasoning ability to determine an optimal and obstacle-free path to carry out its assigned mission.

This technology enables autonomous carts to transfer materials throughout a manufacturing process without the high degree of preparation required with conventional preprogrammed robotic systems. In military situations autonomous vehicles could carry out precise missions while adjusting to rapidly changing environments and battlefield conditions.

Machine vision is also improving the ability of robots to interact with humans. Using small, inexpensive cameras, head- and eye-tracking software can sense where a human user is, allowing robots, as well as virtual personalities on a screen, to maintain eye contact, a key element for natural interactions. Head- and eye-tracking systems have been developed at Carnegie Mellon University and MIT and are offered by small companies such as Seeing Machines of Australia.

An impressive demonstration of machine vision was a vehicle that was driven by an AI system with no human intervention for almost the entire distance from Washington, D.C., to San Diego.[48] Bruce Buchanan,

computer-science professor at the University of Pittsburgh and president of the American Association of Artificial Intelligence, pointed out that this feat would have been "unheard of 10 years ago."

Palo Alto Research Center (PARC) is developing a swarm of robots that can navigate in complex environments, such as a disaster zone, and find items of interest, such as humans who may be injured. In a September 2004 demonstration at an AI conference in San Jose, they demonstrated a group of self-organizing robots on a mock but realistic disaster area.[49] The robots moved over the rough terrain, communicated with one another, used pattern recognition on images, and detected body heat to locate humans.

Speech and language. Dealing naturally with language is the most challenging task of all for artificial intelligence. No simple tricks, short of fully mastering the principles of human intelligence, will allow a computerized system to convincingly emulate human conversation, even if restricted to just text messages. This was Turing's enduring insight in designing his eponymous test based entirely on written language..

Although not yet at human levels, natural language-processing systems are making solid progress. Search engines have become so popular that "Google" has gone from a proper noun to a common verb, and its technology has revolutionized research and access to knowledge. Google and other search engines use AI-based statistical-learning methods and logical inference to determine the ranking of links. The most obvious failing of these search engines is their inability to understand the context of words. Although an experienced user learns how to design a string of keywords to find the most relevant sites (for example, a search for "computer chip" is likely to avoid references to potato chips that a search for "chip" alone might turn up) what we would really like to be able to do is converse with our search engines in natural language. Microsoft has developed a natural-language search engine called Ask MSR (Ask MicroSoft Research), which actually answers natural-language questions such as "When was Mickey Mantle born?"[50] After the system parses the sentence to determine the parts of speech (subject, verb, object, adjective and adverb modifiers, and so on), a special search engine then finds matches based on the parsed sentence. The found documents are searched for sentences that appear to answer the question, and the possible answers are ranked. At least 75 percent of the time, the correct answer is in the top three ranked positions, and incorrect answers are usually obvious (such as "Mickey Mantle was born in 3"). The researchers hope to include knowledge bases that will lower the rank of many of the nonsensical answers.

Microsoft researcher Eric Brill, who has led research on Ask MSR, has also attempted an even more difficult task: building a system that provides answers of about fifty words to more complex questions, such as, "How are the recipients of the Nobel Prize selected?" One of the strategies used by this system is to find an appropriate FAQ section on the Web that answers the query.

Natural-language systems combined with large-vocabulary, speaker-independent (that is, responsive to any speaker) speech recognition over the phone are entering the marketplace to conduct routine transactions. You can talk to British Airways' virtual travel agent about anything you like as long as it has to do with booking flights on British Airways.[51]

You're also likely to talk to a virtual person if you call Verizon for customer service or Charles Schwab and Merrill Lynch to conduct financial transactions. These systems, while they can be annoying to some people, are reasonably adept at responding appropriately to the often ambiguous and fragmented way people speak. Microsoft and other companies are offering systems that allow a business to create virtual agents to book reservations for travel and hotels and conduct routine transactions of all kinds through two-way, reasonably natural voice dialogues.

Not every caller is satisfied with the ability of these virtual agents to get the job done, but most systems provide a means to get a human on the line. Companies using these systems report that they reduce the need for human service agents up to 80 percent. Aside from the money saved, reducing the size of call centers has management benefit. Call-center jobs have very high turnover rates because of low job satisfaction.

It's said that men are loath to ask others for directions, but car vendors are betting that both male and female drivers will be willing to ask their own car for help in getting to their destination. In 2005 the Acura RL and Honda Odyssey will be offering a system from IBM that allows users to converse with their cars.[52] Driving directions will include street names (for example, "turn left on Main Street, then right on Second Avenue"). Users can ask such questions as, "Where is the nearest Italian restaurant?" or they can enter specific locations by voice, ask for clarifications on directions, and give commands to the car itself (such as "turn up the air conditioning"). The Acura RL will also track road conditions and highlight traffic congestion on its screen in real time. The speech recognition is claimed to be speaker-independent and to be unaffected by engine sound, wind, and other noises. The system will reportedly recognize 1.7 million street and city names, in addition to nearly one thousand commands.

Computer language translation continues to improve gradually. Because this is a Turing-level task—that is, it requires full human-level understanding of language to perform at human levels—it will be one of the last application areas to compete with human performance. Franz Josef Och, a computer scientist at the University of Southern California, has developed a technique that can generate a new language-translation system between any pair of languages in a matter of hours or days.[53] All he needs is a "Rosetta stone"—that is, text in one language and the translation of that text in the other language—although he needs millions of words of such translated text. Using a self-organizing technique, the system is able to develop its own statistical models of how text is translated from one language to the other and develops these models in both directions.

This contrasts with other translation systems, in which linguists painstakingly code grammar rules with long lists of exceptions to each rule. Och's system recently received the highest score in a competition of translation systems conducted by the U.S. Commerce Department's National Institute of Standards and Technology.

Entertainment and sports. In an amusing and intriguing application of GAs, Oxford scientist Torsten Reil created animated creatures with simulated joints and muscles and a neural net for a brain. He then assigned them a task: to walk. He used a GA to evolve this capability, which involved seven hundred parameters. "If you look at that system

with your human eyes, there's no way you can do it on your own, because the system is just too complex," Reil points out. "That's where evolution comes in."[54]

While some of the evolved creatures walked in a smooth and convincing way, the research demonstrated a well-known attribute of GAs: you get what you ask for. Some creatures figured out novel new ways of passing for walking. According to Weil, "We got some creatures that didn't walk at all, but had these very strange ways of moving forward: crawling or doing somersaults."

Software is being developed that can automatically extract excerpts from a video of a sports game that show the more important plays.[55] A team at Trinity College in Dublin is working on table-based games like pool, in which software tracks the location of each ball and is programmed to identify when a significant shot has been made. A team at the University of Florence is working on soccer. This software tracks the location of each player and can determine the type of play being made (such as free kicking or attempting a goal), when a goal is achieved, when a penalty is earned, and other key events.

The Digital Biology Interest Group at University College in London is designing Formula One race cars by breeding them using GAs.[56]

The AI winter is long since over. We are well into the spring of narrow AI. Most of the examples above were research projects just ten to fifteen years ago. If all the AI systems in the world suddenly stopped functioning, our economic infrastructure would grind to a halt. Your bank would cease doing business. Most transportation would be crippled. Most communications would fail. This was not the case a decade ago. Of course, our AI systems are not smart enough—yet—to organize such a conspiracy.

Strong AI

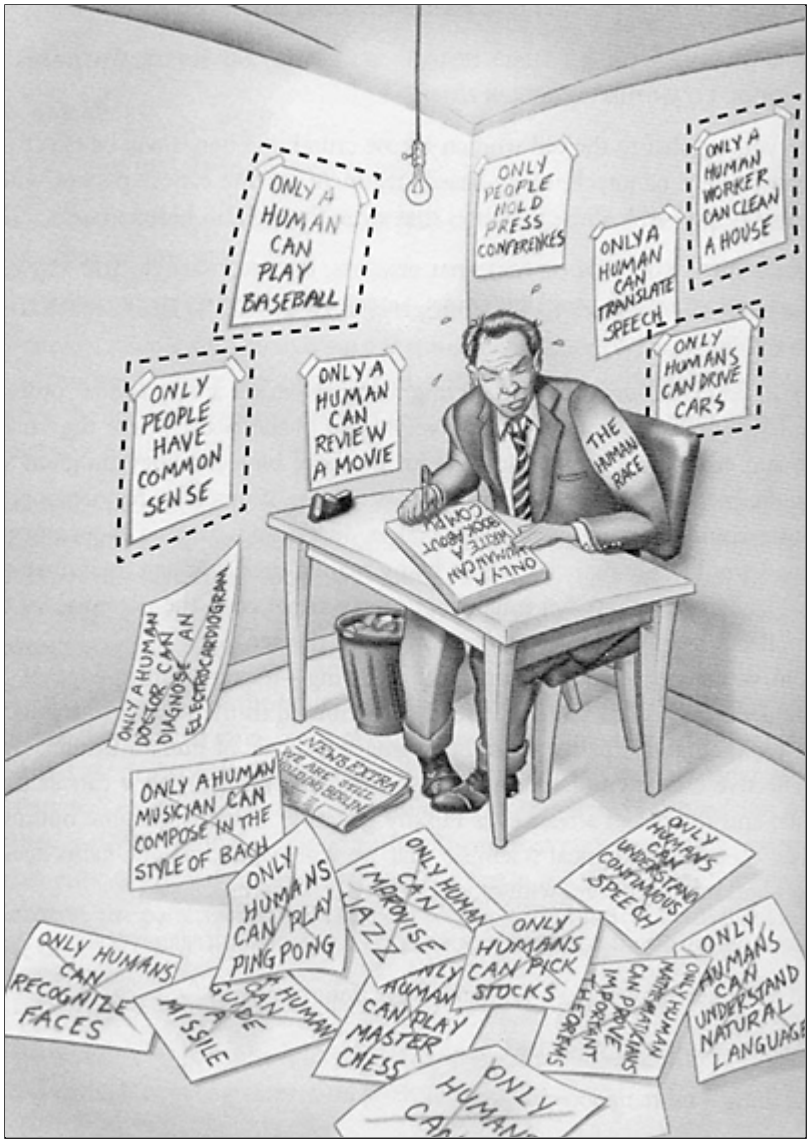
> *If you understand something in only one way, then you don't really understand it at all. This is because, if something goes wrong, you get stuck with a thought that just sits in your mind with nowhere to go. The secret of what anything means to us depends on how we've connected it to all the other things we know. This is why, when someone learns 'by rote,' we say that they don't really understand. However, if you have several different representations then, when one approach fails you can try another. Of course, making too many indiscriminate connections will turn a mind to mush. But well-connected representations let you turn ideas around in your mind, to envision things from many perspectives until you find one that works for you. And that's what we mean by thinking!* —Marvin Minsky[57]

> *Advancing computer performance is like water slowly flooding the landscape. A half century ago it began to drown the lowlands, driving out human calculators and record clerks, but leaving most of us dry. Now the flood has reached the foothills, and our outposts there are contemplating retreat. We feel safe on our peaks, but, at the present rate, those too will be submerged within another half century. I propose that we build Arks as that day nears, and adopt a seafaring life!*

> *For now, though, we must rely on our representatives in the lowlands to tell us what water is really like.*
>
> *Our representatives on the foothills of chess and theorem-proving report signs of intelligence. Why didn't we get similar reports decades before, from the lowlands, as computers surpassed humans in arithmetic and rote memorization? Actually, we did, at the time. Computers that calculated like thousands of mathematicians were hailed as "giant brains," and inspired the first generation of AI research. After all, the machines were doing something beyond any animal, that needed human intelligence, concentration and years of training. But it is hard to recapture that magic now. One reason is that computers' demonstrated stupidity in other areas biases our judgment. Another relates to our own ineptitude. We do arithmetic or keep records so painstakingly and externally, that the small mechanical steps in a long calculation are obvious, while the big picture often escapes us. Like Deep Blue's builders, we see the process too much from the inside to appreciate the subtlety that it may have on the outside. But there is a non-obviousness in snowstorms or tornadoes that emerge from the repetitive arithmetic of weather simulations, or in rippling tyrannosaur skin from movie animation calculations. We rarely call it intelligence, but "artificial reality" may be an even more profound concept than artificial intelligence (Moravec 1998).*
>
> *The mental steps underlying good human chess playing and theorem proving are complex and hidden, putting a mechanical interpretation out of reach. Those who can follow the play naturally describe it instead in mentalistic language, using terms like strategy, understanding and creativity. When a machine manages to be simultaneously meaningful and surprising in the same rich way, it too compels a mentalistic interpretation. Of course, somewhere behind the scenes, there are programmers who, in principle, have a mechanical interpretation. But even for them, that interpretation loses its grip as the working program fills its memory with details too voluminous for them to grasp.*
>
> *As the rising flood reaches more populated heights, machines will begin to do well in areas a greater number can appreciate. The visceral sense of a thinking presence in machinery will become increasingly widespread. When the highest peaks are covered, there will be machines than can interact as intelligently as any human on any subject. The presence of minds in machines will then become self-evident.* —Hans Moravec[58]

Because of the exponential nature of progress in information-based technologies, performance often shifts quickly from pathetic to daunting. In many diverse realms, as the examples in the previous section make clear, the performance of narrow AI is already impressive. The range of intelligent tasks in which machines can now compete with human intelligence is continually expanding. In a cartoon in *The Age of Spiritual Machines*, a defensive "human race" is seen writing out signs that state what only people (and not machines) can do.[59] Littered on the floor are

the signs the human race has already discarded, because machines can now perform these functions: diagnose an electrocardiogram, compose in the style of Bach, recognize faces, guide a missile, play Ping-Pong, play master chess, pick stocks, improvise jazz, prove important theorems, and understand continuous speech. Back in 1999 these tasks were no longer solely the province of human intelligence; machines could do them all.



On the wall behind the man symbolizing the human race were signs he had written out describing the tasks that were still the sole province of humans: have common sense, review a movie, hold press conferences, translate speech, clean a house, and drive cars. If we were to redesign this cartoon in a few years, some of these signs would also be likely to end up on the floor. When CYC reaches one hundred million items of commonsense knowledge, perhaps human superiority in the realm of commonsense reasoning won't be so clear.

The era of household robots, although still fairly primitive today, has already started. Ten years from now, it's likely we will consider "clean a house" as within the capabilities of machines. As for driving cars, robots with no human intervention have already driven nearly across the United States on ordinary roads with other normal traffic. We are not yet ready to turn over all steering wheels to machines, but there are serious

proposals to create electronic highways on which cars (with people in them) will drive by themselves.

The three tasks that have to do with human-level understanding of natural language—reviewing a movie, holding a press conference, and translating speech—are the most difficult. Once we can take down these signs, we'll have Turing-level machines, and the era of strong AI will have started.

This era will creep up on us. As long as there are any discrepancies between human and machine performance—areas in which humans outperform machines—strong AI skeptics will seize on these differences. But our experience in each area of skill and knowledge is likely to follow that of Kasparov. Our perceptions of performance will shift quickly from pathetic to daunting as the knee of the exponential curve is reached for each human capability.

How will strong AI be achieved? Most of the material in this book is intended to lay out the fundamental requirements for both hardware and software and explain why we can be confident that these requirements will be met in nonbiological systems. The continuation of the exponential growth of the price-performance of computation to achieve hardware capable of emulating human intelligence was still controversial in 1999. There has been so much progress in developing the technology for three-dimensional computing over the past five years that relatively few knowledgeable observers now doubt that this will happen. Even just taking the semiconductor industry's published ITRS road map, which runs to 2018, we can project human-level hardware at reasonable cost by that year.[60]

I've stated the case in chapter 4 *The Singularity is Near* of why we can have confidence that we will have detailed models and simulations of all regions of the human brain by the late 2020s. Until recently, our tools for peering into the brain did not have the spatial and temporal resolution, bandwidth, or price-performance to produce adequate data to create sufficiently detailed models. This is now changing. The emerging generation of scanning and sensing tools can analyze and detect neurons and neural components with exquisite accuracy, while operating in real time.

Future tools will provide far greater resolution and capacity. By the 2020s, we will be able to send scanning and sensing nanobots into the capillaries of the brain to scan it from inside. We've shown the ability to translate the data from diverse sources of brain scanning and sensing into models and computer simulations that hold up well to experimental comparison with the performance of the biological versions of these regions. We already have compelling models and simulations for several important brain regions. As I argued in chapter 4 of *The Singularity is Near*, it's a conservative projection to expect detailed and realistic models of all brain regions by the late 2020s.

One simple statement of the strong AI scenario is that we will learn the principles of operation of human intelligence from reverse engineering all the brain's regions, and we will apply these principles to the brain-capable computing platforms that will exist in the 2020s. We already have an effective toolkit for narrow AI. Through the ongoing refinement of these methods, the development of new algorithms, and the trend toward combining multiple methods into intricate architectures, narrow

AI will continue to become less narrow. That is, AI applications will have broader domains, and their performance will become more flexible. AI systems will develop multiple ways of approaching each problem, just as humans do. Most important, the new insights and paradigms resulting from the acceleration of brain reverse engineering will greatly enrich this set of tools on an ongoing basis. This process is well under way.

It's often said that the brain works differently from a computer, so we cannot apply our insights about brain function into workable nonbiological systems. This view completely ignores the field of self-organizing systems, for which we have a set of increasingly sophisticated mathematical tools. As I discussed in chapter 4 of *The Singularity is Near*, the brain differs in a number of important ways from that of conventional, contemporary computers. If you open up your Palm Pilot and cut a wire, there's a good chance you will break the machine. Yet we routinely lose many neurons and interneuronal connections with no ill effect, because the brain is self-organizing and relies on distributed patterns in which many specific details are not important.

When we get to the mid- to late 2020s, we will have access to a generation of extremely detailed brain-region models. Ultimately the toolkit will be greatly enriched with these new models and simulations and will encompass a full knowledge of how the brain works. As we apply the toolkit to intelligent tasks, we will draw upon the entire range of tools, some derived directly from brain reverse engineering, some merely inspired by what we know about the brain, and some not based on the brain at all but on decades of AI research.

Part of the brain's strategy is to learn information, rather than having knowledge hard-coded from the start. ("Instinct" is the term we use to refer to such innate knowledge.) Learning will be an important aspect of AI, as well. In my experience in developing pattern-recognition systems in character recognition, speech recognition, and financial analysis, providing for the AI's education is the most challenging and important part of the engineering. With the accumulated knowledge of human civilization increasingly accessible online, future AIs will have the opportunity to conduct their education by accessing this vast body of information.

The education of AIs will be much faster than that of unenhanced humans. The twenty-year time span required to provide a basic education to biological humans could be compressed into a matter of weeks or less. Also, because nonbiological intelligence can share its patterns of learning and knowledge, only one AI has to master each particular skill. As I pointed out, we trained one set of research computers to understand speech, but then the hundreds of thousands of people who acquired our speech-recognition software had to load only the already trained patterns into their computers.

One of the many skills that nonbiological intelligence will achieve with the completion of the human brain–reverse engineering project is sufficient mastery of language and shared human knowledge to pass the Turing test. The Turing test is important not so much for its practical significance but rather because it will demarcate a crucial threshold. As I have pointed out, there is no simple means to pass a Turing test, other than to convincingly emulate the flexibility, subtlety, and suppleness of human intelligence. Having captured that capability in our technology, it will then be subject to engineering's ability to concentrate, focus, and

amplify it..

Variations of the Turing test have been proposed. The annual Loebner Prize contest awards a bronze prize to the chatterbot (conversational bot) best able to convince human judges that it's human.[61] The criteria for winning the silver prize is based on Turing's original test, and it obviously has yet to be awarded. The gold prize is based on visual and auditory communication. In other words, the AI must have a convincing face and voice, as transmitted over a terminal, and thus it must appear to the human judge as if he or she is interacting with a real person over a videophone. On the face of it, the gold prize sounds more difficult. I've argued that it may actually be easier, because judges may pay less attention to the text portion of the language being communicated and could be distracted by a convincing facial and voice animation. In fact, we already have real-time facial animation, and while it is not quite up to these modified Turing standards, it's reasonably close. We also have very natural-sounding voice synthesis, which is often confused with recordings of human speech, although more work is needed on prosodics (intonation). We're likely to achieve satisfactory facial animation and voice production sooner than the Turing-level language and knowledge capabilities.

Turing was carefully imprecise in setting the rules for his test, and significant literature has been devoted to the subtleties of establishing the exact procedures for determining how to assess when the Turing test has been passed.[62] In 2002 I negotiated the rules for a Turing-test wager with Mitch Kapor on the Long Now Web site.[63] The question underlying our twenty-thousand-dollar bet, the proceeds of which go to the charity of the winner's choice, was, "Will the Turing test be passed by a machine by 2029?" I said yes, and Kapor said no. It took us months of dialogue to arrive at the intricate rules to implement our wager. Simply defining "machine" and "human," for example, was not a straightforward matter. Is the human judge allowed to have any nonbiological thinking processes in his or her brain? Conversely, can the machine have any biological aspects?

Because the definition of the Turing test will vary from person to person, Turing test-capable machines will not arrive on a single day, and there will be a period during which we will hear claims that machines have passed the threshold. Invariably, these early claims will be debunked by knowledgeable observers, probably including myself. By the time there is a broad consensus that the Turing test has been passed, the actual threshold will have long since been achieved.

Edward Feigenbaum proposes a variation of the Turing test, which assesses not a machine's ability to pass for human in casual, everyday dialogue but its ability to pass for a scientific expert in a specific field.[64] The Feigenbaum test (FT) may be more significant than the Turing test, because FT-capable machines, being technically proficient, will be capable of improving their own designs. Feigenbaum describes his test in this way:

Two players play the FT game. One player is chosen from among the elite practitioners in each of three pre-selected fields of natural science, engineering, or medicine. (The number could be larger, but for this challenge not greater than ten). Let's say we choose the fields from among those covered in the U.S. National Academy.... For example, we

could choose astrophysics, computer science, and molecular biology. In each round of the game, the behavior of the two players (elite scientist and computer) is judged by another Academy member in that particular domain of discourse, e.g., an astrophysicist judging astrophysics behavior. Of course the identity of the players is hidden from the judge as it is in the Turing test. The judge poses problems, asks questions, asks for explanations, theories, and so on—as one might do with a colleague. Can the human judge choose, at better than chance level, which is his National Academy colleague and which is the computer?

Of course Feigenbaum overlooks the possibility that the computer might already be a National Academy colleague, but he is obviously assuming that machines will not yet have invaded institutions that today comprise exclusively biological humans. While it may appear that the FT is more difficult than the Turing test, the entire history of AI reveals that machines started with the skills of professionals and only gradually moved towards the language skills of a child. Early AI systems demonstrated their prowess initially in professional fields such as proving mathematical theorems and diagnosing medical conditions. These early systems would not be able to pass the FT, however, because they do not have the language skills and the flexible ability to model knowledge from different perspectives, which are needed to engage in the professional dialogue inherent in the FT.

This language ability is essentially the same ability needed in the Turing test. Reasoning in many technical fields is not necessarily more difficult than the commonsense reasoning engaged in by most human adults. I would expect that machines will pass the FT, at least in some disciplines, around the same time as they pass the Turing test. Passing the FT in all disciplines is likely to take longer, however. This is why I see the 2030s as a period of consolidation, as machine intelligence rapidly expands its skills and incorporates the vast knowledge bases of our biological human and machine civilization. By the 2040s we will have the opportunity to apply the accumulated knowledge and skills of our civilization to computational platforms that are billions of times more capable than unassisted biological human intelligence.

The advent of strong AI is the most important transformation this century will see. Indeed, it's comparable in importance to the advent of biology itself. It will mean that a creation of biology has finally mastered its own intelligence and discovered means to overcome its limitations. Once the principles of operation of human intelligence are understood, expanding its abilities will be conducted by human scientists and engineers whose own biological intelligence will have been greatly amplified through an intimate merger with nonbiological intelligence. Over time, the nonbiological portion will predominate.

I have discussed aspects of the impact of this transformation throughout *The Singularity is Near*, which I focus on in chapter 6. Intelligence is the ability to solve problems with limited resources, including limitations of time. The Singularity will be characterized by the rapid cycle of human intelligence—increasingly nonbiological— capable of comprehending and leveraging its own powers.

---

1 As quoted by Douglas Hofstadter in *Gödel, Escher, Bach: An Eternal Golden Braid* (New York: Basic Books, 1979).

2 The author runs a company, FATKAT (Financial Accelerating Transactions by Kurzweil Adaptive Technologies), which applies computerized pattern recognition to financial data to make stock-market investment decisions, http://www.FatKat.com.

3 See discussion in chapter 2 on price-performance improvements in computer memory and electronics in general.

4 Runaway AI refers to a scenario where, as Max More describes "superintelligent *machines*, initially harnessed for *human* benefit, soon leave us behind." Max More, "Embrace, Don't Relinquish, the Future," http://www.kurzweilai.net/articles/art0106.html?printable=1

See also Damien Broderick's description of the "Seed AI" as "A self-improving seed AI could run glacially slowly on a limited machine substrate. The point is, so long as it has the capacity to improve itself, at some point it will do so convulsively, bursting through any architectural bottlenecks to design its own improved hardware, maybe even build it (if it's allowed control of tools in a fabrication plant)." Damien Broderick, "Tearing toward the Spike," presented at "Australia at the Crossroads? Scenarios and Strategies for the Future," (April 31 - May 2, 2000), published on KurzweilAI.net May 7, 2001: http://www.kurzweilai.net /meme/frame.html?main=/articles/art0173.html

5 David Talbot, "Lord of the Robots," *Technology Review* (April 2002).

6 Heather Havenstein writes that the "inflated notions spawned by science fiction writers about the convergence of humans and machines tarnished the image of AI in the 1980s because AI was perceived as failing to live up to its potential." Heather Havenstein, "Spring comes to AI winter: A thousand applications bloom in medicine, customer service, education and manufacturing," *Computerworld*, February 14, 2005, http://www.computerworld.com/softwaretopics/software/story /0,10801,99691,00.html This tarnished image led to "AI Winter," defined as "a term coined by Richard Gabriel for the (circa 1990-94?) crash of the wave of enthusiasm for the AI language Lisp and AI itself, following a boom in the 1980s." Duane Rettig wrote "...companies rode the great AI wave in the early 80's, when large corporations poured billions of dollars into the AI hype that promised thinking machines in 10 years. When the promises turned out to be harder than originally thought, the AI wave crashed, and Lisp crashed with it because of its association with AI. We refer to it as the AI Winter." Duane Rettig quoted in "AI Winter," http://c2.com/cgi/wiki?AiWinter

7 The General Problem Solver (GPS) computer program, written in 1957, was able to solve problems through rules that allowed the GPS to divide a problem's goals into subgoals, and then check if obtaining a particular subgoal would bring the GPS closer to solving the overall goal. In the early 1960s Thomas Evan wrote ANALOGY, a "program [that] solves geometric-analogy problems of the form A:B::C:? taken from IQ tests and college entrance exams." Boicho Kokinov and Robert M. French, Computational Models of Analogy-Making, in Nadel, L. (Ed.) *Encyclopedia of Cognitive Science*, Vol. 1, (London: Nature Publishing Group, 2003) pp.113-118. See also A. Newell, J.C. Shaw, and H.A. Simon, "Report on a general problem-solving program," *Proceedings of the International Conference on Information Processing*, (Paris: UNESCO House, 1959) pp. 256-264; and Thomas Evans, "A Heuristic Program to Solve Geometric-

Analogy Problems," in *Semantic Information Processing*, M. Minsky, Editor, (Cambridge, MA: MIT Press, 1968).

8 Sir Arthur Conan Doyle, "The Red-Headed League," 1890, available at http://www.eastoftheweb.com/short-stories/UBooks/RedHead.shtml.

9 V. Yu et al., "Antimicrobial Selection by a Computer: A Blinded Evaluation by Infectious Diseases Experts," JAMA 242.12 (1979): 1279–82.

10 Gary H. Anthes, "Computerizing Common Sense," *Computerworld*, April 8, 2002, http://www.computerworld.com/news/2002/story/0,11280,69881,00.html.

11 Kristen Philipkoski, "Now Here's a Really Big Idea," *Wired* News, November 25, 2002, http://www.wired.com/news/technology/0,1282,56374,00.html, reporting on Darryl Macer, "The Next Challenge Is to Map the Human Mind," *Nature* 420 (November 14, 2002): 121; see also a description of the project at http://www.biol.tsukuba.ac.jp/~macer/index.html.

12 Thomas Bayes, "An Essay Towards Solving a Problem in the Doctrine of Chances," published in 1763, two years after his death in 1761.

13 SpamBayes spam filter, http://spambayes.sourceforge.net.

14 Lawrence R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE* 77 (1989): 257–86. For a mathematical treatment of Markov models, see http://jedlik.phy.bme.hu/~gerjanos/HMM/node2.html.

15 Kurzweil Applied Intelligence (KAI), founded by the author in 1982, was sold in 1997 for $100 million and is now part of ScanSoft (formerly called Kurzweil Computer Products, the author's first company, which was sold to Xerox in 1980), now a public company. KAI introduced the first commercially marketed large-vocabulary speech-recognition system in 1987 (Kurzweil Voice Report, with a ten-thousand-word vocabulary).

16 Here is the basic schema for a neural net algorithm. Many variations are possible, and the designer of the system needs to provide certain critical parameters and methods, detailed below.

Creating a neural-net solution to a problem involves the following steps:

- Define the input.
- Define the topology of the neural net (i.e., the layers of neurons and the connections between the neurons).
- Train the neural net on examples of the problem.
- Run the trained neural net to solve new examples of the problem.
- Take your neural-net company public. These steps (except for the last one) are detailed below:

The Problem Input

The problem input to the neural net consists of a series of numbers. This input can be:

- In a visual pattern-recognition system, a two-dimensional array of numbers representing the pixels of an image; or
- In an auditory (e.g., speech) recognition system, a two-dimensional array of numbers representing a sound, in which the first dimension represents parameters of the sound (e.g., frequency components) and the second dimension represents different points in time; or
- In an arbitrary pattern-recognition system, an n-dimensional array of numbers representing the input pattern.

Defining the Topology

To set up the neural net, the architecture of each neuron consists of:

- Multiple inputs in which each input is "connected" to either the output of another neuron, or one of the input numbers.
- Generally, a single output, which is connected either to the input of another neuron (which is usually in a higher layer), or to the final output.

Set up the First Layer of Neurons

- Create N_0 neurons in the first layer. For each of these neurons, "connect" each of the multiple inputs of the neuron to "points" (i.e., numbers) in the problem input. These connections can be determined randomly or using an evolutionary algorithm (see below).
- Assign an initial "synaptic strength" to each connection created. These weights can start out all the same, can be assigned randomly, or can be determined in another way (see below).

Set up the Additional Layers of Neurons

Set up a total of M layers of neurons. For each layer, set up the neurons in that layer. For $layer_i$:

- Create N_i neurons in $layer_i$. For each of these neurons, "connect" each of the multiple inputs of the neuron to the outputs of the neurons in $layer_{i-1}$ (see variations below).
- Assign an initial "synaptic strength" to each connection created. These weights can start out all the same, can be assigned randomly, or can be determined in another way (see below).
- The outputs of the neurons in layerM are the outputs of the neural net (see variations below).

The Recognition Trials

How Each Neuron Works Once the neuron is set up, it does the following for each recognition trial.

- Each weighted input to the neuron is computed by multiplying the output of the other neuron (or initial input) that the input to this neuron is connected to by the synaptic strength of that connection.

- All of these weighted inputs to the neuron are summed.
- If this sum is greater than the firing threshold of this neuron, then this is neuron is considered to fire and its output is 1. Otherwise, its output is 0 (see variations below).

Do the Following for Each Recognition Trial

For each layer, from $layer_0$ to $layer_M$: For each neuron in the layer:

- Sum its weighted inputs (each weighted input = the output of the other neuron [or initial input] that the input to this neuron is connected to multiplied by the synaptic strength of that connection).
- If this sum of weighted inputs is greater than the firing threshold for this neuron, set the output of this neuron = 1, otherwise set it to 0.

To Train the Neural Net

- Run repeated recognition trials on sample problems.
- After each trial, adjust the synaptic strengths of all the interneuronal connections to improve the performance of the neural net on this trial (see the discussion below on how to do this).
- Continue this training until the accuracy rate of the neural net is no longer improving (i.e., reaches an asymptote).

Key Design Decisions

In the simple schema above, the designer of this neural-net algorithm needs to determine at the outset:

- What the input numbers represent.
- The number of layers of neurons.
- The number of neurons in each layer. (Each layer does not necessarily need to have the same number of neurons.)
- The number of inputs to each neuron in each layer. The number of inputs (i.e., interneuronal connections) can also vary from neuron to neuron and from layer to layer.
- The actual "wiring" (i.e., the connections). For each neuron in each layer, this consists of a list of other neurons, the outputs of which constitute the inputs to this neuron. This represents a key design area. There are a number of possible ways to do this:
  - (i) Wire the neural net randomly; or
  - (ii) Use an evolutionary algorithm (see below) to determine an optimal wiring; or
  - (iii) Use the system designer's best judgment in determining the wiring.
- The initial synaptic strengths (i.e., weights) of each connection. There are a number of possible ways to do this:
  - (i) Set the synaptic strengths to the same value; or
  - (ii) Set the synaptic strengths to different random values; or
  - (iii) Use an evolutionary algorithm to determine an

optimal set of initial values; or
    - (iv) Use the system designer's best judgment in determining the initial values.

- The firing threshold of each neuron.
- Determine the output. The output can be:
    - (i) the outputs of layerM of neurons; or
    - (ii) the output of a single output neuron, the inputs of which are the outputs of the neurons in layerM.
    - (iii) a function of (e.g., a sum of) the outputs of the neurons in layerM; or
    - (iv) another function of neuron outputs in multiple layers.

- Determine how the synaptic strengths of all the connections are adjusted during the training of this neural net. This is a key design decision and is the subject of a great deal of research and discussion. There are a number of possible ways to do this:
    - (i) For each recognition trial, increment or decrement each synaptic strength by a (generally small) fixed amount so that the neural net's output more closely matches the correct answer. One way to do this is to try both incrementing and decrementing and see which has the more desirable effect. This can be time-consuming, so other methods exist for making local decisions on whether to increment or decrement each synaptic strength.
    - (ii) Other statistical methods exist for modifying the synaptic strengths after each recognition trial so that the performance of the neural net on that trial more closely matches the correct answer.

Note that neural-net training will work even if the answers to the training trials are not all correct. This allows using real-world training data that may have an inherent error rate. One key to the success of a neural net–based recognition system is the amount of data used for training. Usually a very substantial amount is needed to obtain satisfactory results. Just like human students, the amount of time that a neural net spends learning its lessons is a key factor in its performance.

Variations

Many variations of the above are feasible:

- There are different ways of determining the topology. In particular, the interneuronal wiring can be set either randomly or using an evolutionary algorithm.
- There are different ways of setting the initial synaptic strengths.
- The inputs to the neurons in $layer_i$ do not necessarily need to come from the outputs of the neurons in $layer_{i-1}$. Alternatively, the inputs to the neurons in each layer can come from any lower layer or any layer.
- There are different ways to determine the final output.
- The method described above results in an "all or nothing" (1 or 0) firing called a nonlinearity. There are other

nonlinear functions that can be used. Commonly a function is used that goes from 0 to 1 in a rapid but more gradual fashion. Also, the outputs can be numbers other than 0 and 1.

- The different methods for adjusting the synaptic strengths during training represent key design decisions. The above schema describes a "synchronous" neural net, in which each recognition trial proceeds by computing the outputs of each layer, starting with layer0 through $layer_M$. In a true parallel system, in which each neuron is operating independently of the others, the neurons can operate "asynchronously" (i.e., independently). In an asynchronous approach, each neuron is constantly scanning its inputs and fires whenever the sum of its weighted inputs exceeds its threshold (or whatever its output function specifies).

17 See Chapter 4 for a detailed discussion of brain reverse-engineering. As one example of the progression, S. J. Thorpe writes, "We have really only just begun what will certainly be a long term project aimed at reverse engineering the primate visual system. For the moment, we have only explored some very simple architectures, involving essentially just feed-forward architectures involving a relatively small numbers of layers… In the years to come, we will strive to incorporate as many of the computational tricks used by the primate and human visual system as possible. More to the point, it seems that by adopting the spiking neuron approach, it will soon be possible to develop sophisticated systems capable of simulating very large neuronal networks in real time." S.J. Thorpe et al., "Reverse engineering of the visual system using networks of spiking neurons," Proceedings of the IEEE 2000 International Symposium on Circuits and Systems, IEEE press. IV: 405-408, http://www.sccn.ucsd.edu/~arno/mypapers/thorpe.pdf

18 T. Schoenauer et. al. write, "Over the past years a huge diversity of hardware for artificial neural networks (ANN) has been designed… Today one can choose from a wide range of neural network hardware. Designs differ in terms of architectural approaches, such as neurochips, accelerator boards and multi-board neurocomputers, as well as concerning the purpose of the system, such as the ANN algorithm(s) and the system's versatility… Digital neurohardware can be classified by the:[sic] system architecture, degree of parallelism, typical neural network partition per processor, inter-processor communication network and numerical representation." T. Schoenauer, A. Jahnke, U. Roth and H. Klar, "Digital Neurohardware: Principles and Perspectives," in *Proc. Neuronale Netze in der Anwendung – Neural Networks in Applications* NN'98, Magdeburg, invited paper (February 1998): 101-106, http://bwrc.eecs.berkeley.edu/People/kcamera/neural/papers /schoenauer98digital.pdf. See also:Yihua Liao, "Neural Networks in Hardware: A Survey," 2001, http://ailab.das.ucdavis.edu/~yihua /research/NNhardware.pdf

19 Here is the basic schema for a genetic (evolutionary) algorithm. Many variations are possible, and the designer of the system needs to provide certain critical parameters and methods, detailed below.

The Evolutionary Algorithm

Create N solution "creatures". Each one has:

- A genetic code: a sequence of numbers that characterize a possible solution to the problem. The numbers can represent critical parameters, steps to a solution, rules, etc.
- For each generation of evolution, do the following:
- Do the following for each of the N solution creatures:
- Apply this solution creature's solution (as represented by its genetic code) to the problem, or simulated environment. Rate the solution.
- Pick the L solution creatures with the highest ratings to survive into the next generation.
- Eliminate the (N–L) nonsurviving solution creatures.
- Create (N–L) new solution creatures from the L surviving solution creatures by:

(i) Making copies of the L surviving creatures. Introduce small random variations into each copy; or

(ii) Create additional solution creatures by combining parts of the genetic code (using "sexual" reproduction, or otherwise combining portions of the chromosomes) from the L surviving creatures; or

(iii) Doing a combination of (i) and (ii).

- Determine whether or not to continue evolving: Improvement = (highest rating in this generation)–(highest rating in the previous generation). If Improvement < Improvement Threshold then we're done.
- The solution creature with the highest rating from the last generation of evolution has the best solution. Apply the solution defined by its genetic code to the problem.

Key Design Decisions

In the simple schema above, the designer needs to determine at the outset:

- Key parameters:
  - N
  - L
  - Improvement threshold
- What the numbers in the genetic code represent and how the solution is computed from the genetic code.
- A method for determining the N solution creatures in the first generation. In general, these need only be "reasonable" attempts at a solution. If these first-generation solutions are too far afield, the evolutionary algorithm may have difficulty converging on a good solution. It is often worthwhile to create the initial solution creatures in such a way that they are reasonably diverse. This will help prevent the evolutionary process from just finding a "locally" optimal solution.
- How the solutions are rated.
- How the surviving solution creatures reproduce.

Variations

Many variations of the above are feasible. For example:

- There does not need to be a fixed number of surviving solutions creatures (L) from each generation. The survival rule(s) can allow for a variable number of survivors.
- There does not need to be a fixed number of new solution creatures created in each generation (N–L). The procreation rules can be independent of the size of the population. Procreation can be related to survival, thereby allowing the fittest solution creatures to procreate the most.
- The decision as to whether or not to continue evolving can be varied. It can consider more than just the highest-rated solution creature from the most recent generation(s). It can also consider a trend that goes beyond just the last two generations

20 Sam Williams, "When Machines Breed," August 12, 2004, http://www.salon.com/tech/feature/2004/08/12/evolvable_hardware/index_np.html.

21 Here is the basic scheme (algorithm description) of recursive search. Many variations are possible, and the designer of the system needs to provide certain critical parameters and methods, detailed below.

The Recursive Algorithm

Define a function (program) "Pick Best Next Step." The function returns a value of "SUCCESS" (we've solved the problem) or "FAILURE" (we didn't solve it). If it returns with a value of SUCCESS, then the function also returns the sequence of steps that solved the problem. Pick Best Next Step does the following:

- Determine if the program can escape from continued recursion at this point. This bullet, and the next two bullets deal with this escape decision. First, determine if the problem has now been solved. Since this call to Pick Best Next Step probably came from the program calling itself, we may now have a satisfactory solution. Examples are:
  - (i) In the context of a game (for example, chess), the last move allows us to win (such as checkmate).
  - (ii) In the context of solving a mathematical theorem, the last step proves the theorem.
  - (iii) In the context of an artistic program (for example, a computer poet or composer), the last step matches the goals for the next word or note.

If the problem has been satisfactorily solved, the program returns with a value of "SUCCESS" and the sequence of steps that caused the success.

- If the problem has not been solved, determine if a solution is now hopeless. Examples are:
  - (i) In the context of a game (such as chess), this move causes us to lose (checkmate for the other side).
  - (ii) In the context of solving a mathematical theorem,

this step violates the theorem.
    - (iii) In the context of an artistic creation, this step violates the goals for the next word or note.
- If the solution at this point has been deemed hopeless, the program returns with a value of "FAILURE."
- If the problem has been neither solved nor deemed hopeless at this point of recursive expansion, determine whether or not the expansion should be abandoned anyway. This is a key aspect of the design and takes into consideration the limited amount of computer time we have to spend. Examples are:
    - (i) In the context of a game (such as chess), this move puts our side sufficiently "ahead" or "behind." Making this determination may not be straightforward and is the primary design decision. However, simple approaches (such as adding up piece values) can still provide good results. If the program determines that our side is sufficiently ahead, then Pick Best Next Step returns in a similar manner to a determination that our side has won (that is, with a value of "SUCCESS"). If the program determines that our side is sufficiently behind, then Pick Best Next Step returns in a similar manner to a determination that our side has lost (that is, with a value of "FAILURE").
    - (ii) In the context of solving a mathematical theorem, this step involves determining if the sequence of steps in the proof is unlikely to yield a proof. If so, then this path should be abandoned, and Pick Best Next Step returns in a similar manner to a determination that this step violates the theorem (that is, with a value of "FAILURE"). There is no "soft" equivalent of success. We can't return with a value of "SUCCESS" until we have actually solved the problem. That's the nature of math.
    - (iii) In the context of an artistic program (such as a computer poet or composer), this step involves determining if the sequence of steps (such as the words in a poem, notes in a song) is unlikely to satisfy the goals for the next step. If so, then this path should be abandoned, and Pick Best Next Step returns in a similar manner to a determination that this step violates the goals for the next step (that is, with a value of "FAILURE").

- If Pick Best Next Step has not returned (because the program has neither determined success nor failure nor made a determination that this path should be abandoned at this point), then we have not escaped from continued recursive expansion. In this case, we now generate a list of all possible next steps at this point. This is where the precise statement of the problem comes in:
    - (i) In the context of a game (such as chess), this involves generating all possible moves for "our" side for the current state of the board. This involves a straightforward codification of the rules of the game.
    - (ii) In the context of finding a proof for a mathematical theorem, this involves listing the

possible axioms or previously proved theorems that can be applied at this point in the solution.
- (iii) In the context of a cybernetic art program, this involves listing the possible words/notes/line segments that could be used at this point. For each such possible next step:
  - (i) Create the hypothetical situation that would exist if this step were implemented. In a game, this means the hypothetical state of the board. In a mathematical proof, this means adding this step (for example, axiom) to the proof. In an art program, this means, adding this word/note/line segment.
  - (ii) Now call Pick Best Next Step to examine this hypothetical situation. This is, of course, where the recursion comes in because the program is now calling itself.
  - (iii) If the above call to Pick Best Next Step returns with a value of "SUCCESS," then return from the call to Pick Best Next Step (that we are now in) also with a value of "SUCCESS." Otherwise consider the next possible step.

If all the possible next steps have been considered without finding a step that resulted in a return from the call to Pick Best Next Step with a value of "SUCCESS," then return from this call to Pick Best Next Step (that we are now in) with a value of "FAILURE."

END OF PICK BEST NEXT STEP

If the original call to Pick Best Next Move returns with a value of "SUCCESS," it will also return the correct sequence of steps:

- (i) In the context of a game, the first step in this sequence is the next move you should make.
- (ii) In the context of a mathematical proof, the full sequence of steps is the proof.
- (iii) In the context of a cybernetic art program, the sequence of steps is your work of art.

If the original call to Pick Best Next Step is "FAILURE," then you need to go back to the drawing board.

Key Design Decisions

In the simple schema above, the designer of the recursive algorithm needs to determine the following at the outset:

- The key to a recursive algorithm is the determination in Pick Best Next Step of when to abandon the recursive expansion. This is easy when the program has achieved clear success (such as checkmate in chess or the requisite solution in a math or combinatorial problem) or clear failure. It is more difficult when a clear win or loss has not yet been achieved. Abandoning a line of inquiry before a

well-defined outcome is necessary because otherwise the program might run for billions of years (or at least until the warranty on your computer runs out).

- The other primary requirement for the recursive algorithm is a straightforward codification of the problem. In a game like chess, that's easy. But in other situations, a clear definition of the problem is not always so easy to come by

22 See Kurzweil Cyberart, http://www.KurzweilCyberArt.com, for further description of Ray Kurzweil's Cybernetic Poet and to download a free copy of the program. See U.S. Patent # 6,647,395 "Poet Personalities," inventors: Ray Kurzweil and John Keklak. Abstract: "A method of generating a poet personality including reading poems, each of the poems containing text, generating analysis models, each of the analysis models representing one of poems and storing the analysis models in a personality data structure. The personality data structure further includes weights, each of the weights associated with each of the analysis models. The weights include integer values."

23 Ben Goertzel, *The Structure of Intelligence*, 1993, Springer-Verlag. Ben Goertzel, *The Evolving Mind*, 1993, Gordon and Breach. Ben Goertzel, *Chaotic Logic*, 1994, Plenum. Ben Goertzel, *From Complexity to Creativity*, 1997, Plenum. For a link to Ben Goertzel's books and essays, see http://www.goertzel.org/work.html

24 KurzweilAI.net (http://www.KurzweilAI.net) provides hundreds of articles by one hundred "big thinkers" and other features on "accelerating intelligence." The site offers a free daily or weekly newsletter on the latest developments in the areas covered by this book. To subscribe, enter your e-mail address (which is maintained in strict confidence and is not shared with anyone) on the home page.

25 John Gosney, Business Communications Company, "Artificial Intelligence: Burgeoning Applications in Industry," June 2003, http://www.bccresearch.com/comm/G275.html.

26 Kathleen Melymuka, "Good Morning, Dave . . . ," *Computerworld*, November 11, 2002, http://www.computerworld.com/industrytopics /defense/story/0,10801,75728,00.html.

27 JTRS Technology Awareness Bulletin, August 2004, http://jtrs.army.mil/sections/technicalinformation /fset_technical.html?tech_aware_2004-8.

28 Otis Port, Michael Arndt, and John Carey, "Smart Tools," spring 2003, http://www.businessweek.com/bw50/content/mar2003/a3826072.htm.

29 Wade Roush, "Immobots Take Control: From Photocopiers to Space Probes, Machines Injected with Robotic Self-Awareness Are Reliable Problem Solvers," *Technology Review* (December 2002/January 2003), http://www.occm.de/roush1202.pdf.

30 Jason Lohn quoted in NASA news release "NASA 'Evolutionary' Software Automatically Designs Antenna," http://www.nasa.gov /lb/centers/ames/news/releases/2004/04_55AR.html

31 Robert Roy Britt, "Automatic Astronomy: New Robotic Telescopes See and Think," June 4, 2003, http://www.space.com/businesstechnology

/technology/automated_astronomy_030604.html.

32 H. Keith Melton, "Spies in the Digital Age," http://www.cnn.com /SPECIALS/cold.war/experience/spies/melton.essay.

33 "United Therapeutics (UT) is a biotechnology company focused on developing chronic therapies for life threatening conditions in three therapeutic areas: cardiovascular, oncology and infectious diseases" (http://www.unither.com). Kurzweil Technologies is working with UT to develop pattern recognition–based analysis from either "Holter" monitoring (twenty-four-hour recordings) or "Event" monitoring (thirty days or more).

34 Kristen Philipkoski, "A Map That Maps Gene Functions," *Wired* News, May 28, 2002, http://www.wired.com/news/medtech /0,1286,52723,00.html.

35 Jennifer Ouellette, "Bioinformatics Moves into the Mainstream," *The Industrial Physicist* (October/November 2003), http://www.sciencemasters.com/bioinformatics.pdf.

36 Port, Arndt, and Carey, "Smart Tools."

37 "Protein Patterns in Blood May Predict Prostate Cancer Diagnosis," National Cancer Institute, October 15, 2002, http://www.nci.nih.gov /newscenter/ProstateProteomics, reporting on E. F. Petricoin et al., "Serum Proteomic Patterns for Detection of Prostate Cancer," *Journal of the National Cancer Institute* 94 (2002): 1576–78.

38 Charlene Laino, "New Blood Test Spots Cancer," December 13, 2002, http://my.webmd.com/content/Article/56/65831.htm; Emanuel F. Petricoin III et al., "Use of Proteomic Patterns in Serum to Identify Ovarian Cancer," *The Lancet* 359.9306 (February 16, 2002): 572–77.

39 Mark Hagland, "Doctor's Orders," January 2003, http://www.healthcare- informatics.com/issues/2003/01_03/cpoe.htm.

40 Ross D. King et al., "Functional Genomic Hypothesis Generation and Experimentation by a Robot Scientist," *Nature* 427 (January 15, 2004): 247–52.

41 Port, Arndt, and Carey, "Smart Tools."

42 "Future Route Releases AI-Based Fraud Detection Product," August 18, 2004, http://www.finextra.com/fullstory.asp?id=12365.

43 John Hackett, "Computers Are Learning the Business," *Collections World,* April 24, 2001, http://www.creditcollectionsworld.com /news/042401_2.htm.

44 "Innovative Use of Artificial Intelligence, Monitoring NASDAQ for Potential Insider Trading and Fraud," AAAI press release, July 30, 2003, http://www.aaai.org/Pressroom/Releases/release-03-0730.html.

45 "Adaptive Learning, Fly the Brainy Skies," *Wired* News, March 2002, http://www.wired.com/wired/archive/10.03/everywhere.html?pg=2.

46 Introduction to Artificial Intelligence, EL 629, Maxwell Air Force Base, Air University Library course www.au.af.mil/au/aul/school/acsc/ai02.htm.

47 See www.Seegrid.com.

48 No Hands Across America Web site, http://cart.frc.ri.cmu.edu/users /hpm/project.archive/reference.file/nhaa.html, and "Carnegie Mellon Researchers Will Prove Autonomous Driving Technologies During a 3,000 Mile, Hands-off-the-Wheel Trip from Pittsburgh to San Diego," Carnegie Mellon press release, http://www- 2.cs.cmu.edu/afs/cs/user/tjochem /www/nhaa/official_press_release.html; Robert J. Derocher, "Almost Human," September 2001, http://www.insight-mag.com/insight/01/09 /col-2-pt-1-ClickCulture.htm.

49 "Search and Rescue Robots," Associated Press, September 3, 2004, http://www.smh.com.au/articles/2004/09 /02/1093939058792.html?oneclick=true.

50 "From Factoids to Facts," *The Economist,* August 26, 2004, http://www.economist.com/science/displayStory.cfm?story_id=3127462.

51 Joe McCool, "Voice Recognition, It Pays to Talk," May 2003, http://www.bcs.org/BCS/Products/Publications/JournalsAndMagazines /ComputerBulletin/OnlineArchive/ may03/voicerecognition.htm.

52 John Gartner, "Finally a Car That Talks Back," *Wired* News, September 2, 2004, http://www.wired.com/news/autotech /0,2554,64809,00.html?tw=wn_14techhead.

53 "Computer Language Translation System Romances the Rosetta Stone," Information Sciences Institute, USC School of Engineering (July 24, 2003), http://www.usc.edu/isinews/stories/102.html.

54 Torsten Reil quoted in Steven Johnson, "Darwin in a Box," *Discover Magazine* 24.8 (August 2003), http://www.discover.com/issues/aug-03 /departments/feattech/

55 "Let Software Catch the Game for You," July 3, 2004, http://www.newscientist.com/news/news.jsp?id=ns99996097.

56 Michelle Delio, "Breeding Race Cars to Win," *Wired* News, June 18, 2004, http://www.wired.com/news/autotech/0,2554,63900,00.html.

57 Marvin Minsky, *The Society of Mind* (New York: Simon & Schuster, 1988).

58 Hans Moravec, "When will computer hardware match the human brain?" J*ournal of Evolution and Technology,* 1998. Volume 1.

59 Ray Kurzweil, *The Age of Spiritual Machines* (New York: Viking, 1999), p. 156.

60 See Chapter 2 endnotes 22 and 23 on the International Technology Roadmap for Semiconductors.

61 "The First Turing Test," http://www.loebner.net/Prizef/loebner-prize.html.

62 Douglas R. Hofstadter, "A Coffeehouse Conversation on the Turing Test," May 1981, included in Ray Kurzweil, *The Age of Intelligent Machines* (Cambridge, Mass.: MIT Press, 1990), pp. 80–102, http://www.kurzweilai.net/meme/frame.html?main=/articles

/art0318.html.

63 Ray Kurzweil "Why I Think I Will Win." And Mitch Kapor, "Why I Think I Will Win." Rules: http://www.kurzweilai.net/meme/frame.html?main=/articles/art0373.html; Kapor: http://www.kurzweilai.net/meme/frame.html?main=/articles/art0412.html; Kurzweil: http://www.kurzweilai.net/meme/frame.html?main=/articles/art0374.html; Kurzweil "final word": http://www.kurzweilai.net/meme/frame.html?main=/articles/art0413.html.

64 Edward A. Feigenbaum, "Some Challenges and Grand Challenges for Computational Intelligence," *Journal of the Association for Computing Machinery* 50 (January 2003): 32–40.