How Evolution Programs Thought

Eric B. Baum

God has chosen the world that is the most perfect, that is to say, the one that is at the same time the simplest in hypotheses and the richest in phenomena.
-- Gottfried Von Liebniz

Since Pythagoras, science has viewed the universe, for all its apparent complexity, as arising from an underlying simplicity, from simple essential elements, like atoms, behaving according to succinct mathematical laws. The nature of mathematical laws is that they determine what will happen. Moreover, our brains and bodies are of the material world. So philosophers from classical times to the present have argued whether the mind and all thoughts can be understood as arising from the deterministic outcome of mathematical processes.

This discussion solidified in 1937, when Alan Turing asked whether one could specify a machine capable of simulating in detail the working of a thinking brain. His answer, now called the Turing machine, stands as the theoretical model of computation. Turing showed that, just as matter results from interaction of simple atoms, computations also can be broken down into simple components: every computation can be realized as a series of applications of a small set of rules involving only 1's and 0's. Such elemental computations are called "syntactic" because they depend only on the form of the rule. Execution of syntactic rules depends only on recognizing symbols, which can be done automatically, not on attributing any meaning to the symbols. Turing argued persuasively that, if equipped with a list of 1's and 0's called a program that specified the order in which such syntactic rules were applied, an ordinary desktop computer could perform any possible computation. Thus an ordinary computer, if it had a sufficiently fast clock rate, sufficient memory, and most crucially the right software, could solve the mathematical equations underlying the physics of your brain, and predict everything you say and think, reducing thought to a series of elemental syntactical operations.

Turing's result was a monumental accomplishment in that it gave scientists a precise language that has the generality to describe mind-- the language of computer programs. This can boggle intuition. Brain processes are physical, thoughts are spiritual, somehow other-worldly, what relationship does either have to a computation? Even computer, cognitive, and neuro- scientists sometimes seem fuzzy on the subject. The answer is that computation is a language capable of describing thought formally. Any possible brain process, and thus presumably any possible thought, corresponds precisely to a particular computation. But it turns out that the computations that are possible obey mathematical laws. Thus by studying computer science, one can hope to derive universal principles that allow one to understand thought just as by studying calculus physicists have advanced our understanding of gravity and mechanics.

To be sure, neither Turing nor anyone since has produced a proof that a computer could simulate the brain-- all we have are compelling arguments. If we scientists can't find a plausible theory explaining all aspects of mind while assuming the computation thesis, we should thus be prepared to reject it and look more broadly. But if we can find such a model that agrees with the data, science will understand mind as firmly and in the same way as we understand any other phenomenon.

At least three aspects of mind have been proposed as counterexamples, violating the computational picture. Philosophers such as David Chalmers argue that the sensation of experience can not arise in a simple machine. Where in a machine is the particular experience you feel when sniffing a rose? A second question is: What is meaning? Your thoughts mean something, they are about events or objects in the world. The philosopher David Searle argues that "syntax is not sufficient for semantics" and thus there must be something more to thought than can be captured in any computer program. Finally, in his 1972 book "What Computers Can't Do", Hubert Dreyfus remarked that all extent computer programs display nothing like human understanding. With the exception of very limited domains such as chess, this remains true today. Is "understanding" some quantity that divides thought from computation?

I argue below that recent research in computational learning theory can plausibly be extended to a model explaining all aspects of thought, subjective and objective, in particular answering these three questions. The proposal is straightforward, consistent with a vast range of data, brings results from the theory of computer science to bear on biological, psychological, and philosophical questions, and makes empirical predictions. It also casts light on the phenomenal success of evolution in designing biology.

Occam's Razor

In the fourteenth century, Willem of Occam codified the principle now known as Occam's razor, that one should choose a simple hypothesis to explain one's data. This principle underlies both scientific and day to day reasoning.

Say you discover that a vase in your dining room is broken. You question your child, and he comes up with an explanation, other than himself. As you ask more questions, he elaborates with additional explanations: the reason a pellet from his sling shot is sitting in amidst the rubble is because it stuck in the crack in somebody's shoe sole and got tracked in earlier. And so on. Although each item he suggests may be plausible, you will prefer the simple theory, because he could have come up with a theory of equal complexity to explain any possible observations.

The same principle caused Galileo to favor Copernicus's simple hypothesis over Ptolemy's complex one. Both theories explained the astronomical data (Ptolemy more accurately than Copernicus), but Ptolemy's required separate elaboration for each planet, and so could explain almost any possible data. By contrast Copernicus's was simple, and so made restricted predictions.

Computer scientists and statisticians studying machine learning and pattern recognition have recently formalized Occam's razor by supplying mathematical definitions of simplicity and showing that simplicity leads to generalization. How might a machine learn a concept, such as "chair" from a collection of labeled pictures, some containing chairs and some not? The goal is a program able to recognize whether any new picture contains a chair of any type, whether beanbag or kitchen, from any vantage point. The machine might sort through some class of possible hypotheses, and find one that explains the data it has seen, but will the hypothesis be correct on previously unseen images? The answer that has emerged is: if the machine finds a very simple hypothesis that explains the data, and if the data is randomly chosen, then the hypothesis will correctly classify new images it was not trained on.

To understand this, consider the following. Say before the machine saw any data, it proposed a hypothesis: a computer program it would use to classify images. We give it 1000 randomly drawn images, and the program correctly says which are of chairs and which aren't. Of course, you'd then expect the program to be right on most additional examples. It wouldn't get the first 1000 tests right by accident, unless it was almost entirely correct in general.

Say instead, the machine had 10 possible programs to choose from, we give it the data, and it turns out one of them gets all 1000 images right. Still, this couldn't be luck, that program must be correct. It also will correctly classify most new examples.

Now say the machine had a language from which to build a hypothesis. As it sees the data, it searches through hypotheses expressed in its language, in a fixed order, and settles on the first program that is correct. If it searches through some astronomical number of programs before getting to that one, there may be nothing special about it. If its language is powerful enough, after all, it would eventually find an explanation for anything. But if the program is one of the first 100, it couldn't have happened by chance. For that program to have succeeded on all 1000 random examples, it must be that the program exploits structure in the examples, and will correctly predict on most future examples which contains a chair.

This ordering of hypotheses is a definition of simplicity. By adopting a programming language, and searching for the shortest program explaining our data, we can exploit Occam's razor. If we find a short enough program correctly classifying our data, we can expect with extremely high probability that it will continue to classify future data. The point is, no such simple hypothesis would exist unless the hypothesis exploited some simple underlying structure of the process producing the data. By discovering simple hypotheses, we can uncover and exploit simple structure in the world that may not have been apparent, as Copernicus did.

The converse of this result, that one must form a concise explanation in order to generalize, is also (roughly speaking) true. What comes next in the sequence 2,4,6,8,10,12,? Except for a preference for simple hypotheses, there is no logical basis to prefer 14 over any other number. A particular learning algorithm that specifies which hypotheses are preferred is called an "inductive bias". With a powerful inductive bias, one can learn, but what one can learn is no longer fully general.

Since computer scientists have understood this for more than 20 years, (early important theorems were proved by Vapnik and Chervonenkis in 1971) why is it that we don't yet have programs that can, for example, recognize objects like chairs in images? The answer is: the problem of finding concise descriptions of data turns out to be computationally hard. Complexity theorists have shown that for interesting cases, the problem is at least NP-complete, a class that indicates extensive computation is required to find a solution.

There are 2^{1000} possible ways of classifying a set of 1000 images (each said to have a chair or not a chair). 2^{1000} is about the same size as the number represented in decimal notation by a 1 followed by 300 zeros. So if, as we search through possible programs in order, the trillion trillionth program correctly classifies the data, that would still be a highly significant result, indicating the program would classify most future images correctly. But we don't have time to search through even a trillion programs, even with the fastest computers available.

We might hope that there is a smarter way to find the shortest program agreeing with the data than by such exhaustive search. A famous conjecture of complexity theory (generally dubbed "P != NP") asserts that there is no very fast shortcut, that there is no fast way of solving NP-complete problems. There will be faster ways than exhaustive search, but in general they will still take astronomical effort. While this is an unproven conjecture, it (like Turing's hypothesis) is supported by such substantial evidence that it is presumed true. This is a fundamental law of computation that we will meet again in understanding mind.

The reader may be familiar with the Zip utility, in common use to compress computer files. Zip embodies a simple algorithm that runs rapidly, but is largely oblivious to the file being compressed. Generalization, by contrast, requires extracting the simple underlying structure particular to a given data set thus achieving a much smaller compression, and can not be done rapidly in such a generic fashion. It is precisely at the point where data is compressed beyond what is easy that the underlying structure becomes apparent and meaningful generalization begins, precisely because that is the point where one must be sensitive to specific, surprisingly compact structure of the particular process producing the data.

These results answer Dreyfus's question of why extant Artificial Intelligence(AI) programs do not display understanding. Human programmers do not have the ability to solve the hard computational problems involved in finding compact hypotheses any more than they can manually solve other NP-complete problems. Thus the AI programs humans write are typically too large to exhibit understanding. However, there are partial exceptions. Chess is an example of a domain where humans have been able to craft very compact programs. The essence of chess programs like Deep Blue is contained in an extremely compact search procedure (10 lines or so of computer code) and a single summation of material balance (9 points for a queen, 5 for a rook, and so on). Such chess programs do seem to understand their limited domain fairly well.

Occam's Razor Extended to Thought

As mentioned above, various theorems guarantee that Occam's razor is necessary and sufficient for generalization in concept learning, defined as the problem of predicting whether a given image (say) is or is not an example of a particular concept like a chair. But thought encompasses much more varied phenomena than just classification.

We can model thought more generally if we conjecture a simple extrapolation of Occam's razor from concept learning to behavior. Consider a robot that interacts with a complex world: sensing, computing what to do, and acting. If the robot interacted with as much sophistication as you do, engaging in computations able to solve a vast variety of problems, including solving new problems and new types of problems as well and flexibly as you would, it would be difficult to say it wasn't thinking. The extended Occam's razor states that if the robot relies on a short enough program and is successful at solving enough problems posed by the world, then it will continue solving new problems posed by the same world with high probability.

The idea is the same as above. Such a short program would not have worked on so many previous problems by luck; the fact that it exists means that it exploits the simple underlying structure of the world, in which case it will continue to do so as new problems arise. The only thing that is different is that the program is now doing something more powerful: it is doing computations that exploit the structure. This goes beyond simply finding a short description of the world; in fact exploiting compact structure is a separate hard computational problem from finding it.

Take, for a simple pedagogical example, Rubik's cube. Rubik's cube is difficult in part because it can take more than 43 quintillion different possible configurations. It is interesting, however, because it has a compact structure: all of these many configurations are generated by a simple cube that can be very compactly described. Handed the cube, it is relatively easy to describe it compactly. Given the description, however, it is plainly still a computational challenge to understand how to solve it.

How could a robot learn to solve Rubik's cube? Asked to solve starting from one or a few starting configurations, and allowed to use relatively large program, the robot could just memorize the solution to those particular instances, in which case it would know nothing about solving from new positions. But if you demand the robot use a very short program to solve a large collection of instances without substantial search, the only way will be for it to find a program exploiting the structure of the problem, in which case it will generalize to solve other instances as well.

Programs can be compact and still very powerful if they are composed of modules corresponding to real concepts, i.e. exploiting underlying structure. For example, say you find a module that knows a sequence of twists that move around just a few of the little cubies, leaving the position of all the rest unchanged. The reason why Rubik's cube is hard is that almost any sequence of actions moves large numbers of cubies, so that efforts to solve one portion of the cube destroy progress previously made elsewhere. Given a few such modules that perform simpler actions, however, the problem is factored. By reusing such modules in different ways, a compact program can solve all instances.

Igor Durdanovic and I were able to illustrate these ideas by evolving compact modular programs to solve several problem domains. To evolve a modular program, we simulated an artificial economy of agents, each of which was initially a random computer program. The agents were "paid" for collaborating on solution of problem instances. New agents were created by a random mutation process and injected into the economy, and agents that lost money were removed. The economic structure promoted evolution of cooperation among the agents and a division of labor arose, in which different agents performed different tasks.

We started such systems on relatively simple problems, for example Rubik's cubes that had been only slightly scrambled, and as they solved these worked up to harder instances. In this way we were able to evolve compact modular programs that knew how to solve about half a Rubik's cube, more than many humans achieve. This code thus generalized to solve vast numbers of cube configurations the program had not been trained on. We were even more successful in other problem domains. For example, we evolved a 5 line program able to solve all block stacking problems of a certain type, even though there are infinitely many, and even though similar problems have long been a hard challenge for AI planning systems.

These experiments also illustrated some interesting aspects of the evolution of cooperation, in programs, the economy, and the ecosystem. Cooperation evolved when a natural law of property rights was imposed on the economy, and didn't otherwise, for intuitively explainable reasons, beyond the scope of this article.

The claim then is that thought is execution of such a modular program. You have modules in your mind that know how to divide the world up and exploit its underlying structure. This exploitation is what we call meaning. This meaning arises from pure syntax through Occam's razor-- the program is so tightly constrained that it acquires meaning. When you plan, when you learn, when you think, this code is executing in appropriate ways. Of course, the modules in your mind are vastly more sophisticated and powerful than any we were able to evolve, in a week or so on a desktop. But they arise and generalize to new problems for a similar reason: Occam's razor applied in the context of a world with simple underlying structure.

In 1980, Lakoff and Johnson remarked on the extent to which metaphor pervades language. Consider for example how the idiom "time is money" is worked into the language. We borrow time, spend time, waste time, invest time, etc. Or consider the metaphor, an argument is a structure. You may feel my argument rests on shaky foundations and needs to be shored up, or that it is buttressed by solid facts. Such metaphors are not only ubiquitous, but combined coherently, grounding abstract thoughts (like reasoning about time) in concrete quantities (like money, which you can hold in your hand). The extended Occam hypothesis explains how this arises. A program can be both short and powerful only by reusing subroutines. The program underlying mind is so short and powerful, that its code is necessarily organized in a modular structure, with modules corresponding to real concepts and reusable in different combinations and contexts. When we think about time, we reuse a module for valuable resource management.

The Evolution of Learning

The natural candidate for the Occam program is the genome. The genome is extraordinarily compact. Its functioning core (after the so-called "junk" is stripped away) is believed to be smaller than the source code for Microsoft Office. The brain, by contrast, is 100 million times bigger. Moreover the genome encodes, in a sense, the results of an extraordinary amount of computation: I estimate that some 10^{35} creatures have lived and died, each contributing in a small measure to its evolution.

The proposal of the genome as the Occam program is controversial, because many psychologists, neural network practitioners, and philosophers have argued that infants are born in a state of tabula rasa, a blank slate from which we acquire knowledge by learning. Recall, however, that learning theory requires one have an inductive bias to learn. In my picture the genome encodes this inductive bias, programs that execute, interacting with sensory data, to learn. The learning so innately programmed appears quite automatic, reliably resulting in creatures with similar abilities provided that they are allowed interaction with the world during development.

Complexity theory tells us that learning is a hard problem, requiring vast computation to extract structure. Yet we learn so fast that we do not have time to do the requisite computation. This is possible only because creatures are preprogrammed to extract meaning. The bulk of the requisite computation, and thus the guts of the process from the point of view of complexity theory, went into the evolution of the genome.

Empirical evidence shows that creatures are in fact programmed with specific inductive biases. If a rat is shocked once at a specific point in its maze, it will avoid that corner. If a rat is sickened only once after eating a particular food, it will never eat that type of food again. However it is difficult to train a rat to avoid a location by making it sick or to avoid a type of food by shocking it. The rat is innately programmed to learn particular behaviors from particular stimuli. Similar results hold true for people. A wealth of evidence supports the view that children learn grammar rapidly and almost automatically, because of strong inductive bias built into their brain by their genome. Moreover, while humans can learn "meaningful" facts from a single presentation, they would find it almost impossible to learn things they are not programmed to recognize as meaningful. This meaning is, in my picture, defined by underlying programs, largely coded into the genome, that exploit underlying structure in the world.

Consider the development of visual cortex. The brain uses parallax to calculate the depth of objects. This calculation must be tuned to the width between the eyes. The DNA program is the same in every cell, but its execution differs from cell to cell as the chemical environment differs. As the brain develops, and the skull grows, the DNA program automatically adjusts brain wiring to compute depth. How did this evolve? DNA programs that better develop function in the chemical environment were preferentially selected. But the chemical environment in cortex includes neural firings stimulated by the sensory flow. Thus a program evolved that, in effect, learns the distance between the eyes. Similarly, the same DNA programs cells to develop into visual cortex or auditory cortex depending on the ambient environment of the developing cell. Similar mechanisms would explain development of more abstract thought, for example, Seay and Harlow's famous discovery that monkeys can only acquire normal social behavior during a critical period in development. Critical periods are a flag indicating genomic programming, as many aspects of development are carefully timed. The DNA program exploits the sensory stream (reflected in the chemical environment of developing cells) to grow appropriate neural structures. Learning and development are two sides of the same coin, and so we should expect evolution of tailored learning programs.

If the genomic program encodes development of modules for a variety of meaningful concepts, everything from valuable resource management and understanding 2-topology to causal reasoning and grammar learning, this should be manifested through local variations of gene expression in developing brains. The technology of gene chips allows measurement of which genes are expressed in which brain region. However producing a valid gene chip currently requires RNA collected from about a million cells. A whole bee brain, which encodes most of these different abilities, is only a million cells. Thus gene chip experiments to date may be averaging over different regions, obscuring the effect. Last year, Gray et al. published an image in Science magazine produced with an alternative technique that seemed to show quite detailed structure in gene expression. I predict that as the technology improves, gene expression data will show fine structure in developing brains and suggest looking at neonatal elephant brains (rather than mice as are generally studied) where it may be easier to measure because subregions are expected to be larger.

The Evolution of Evolution

It is also worth noting that evolution itself develops and employs powerful computational modules that exploit meaning. For example, the genome encodes names invoking compact computational subroutines, gene networks, that encode meaningful quantities such as developing a leg or an extension of a leg or an antennae. Expression of the single gene ey on a fly wing during development, for example, will cause a well formed eye to grow there. The genome encodes Hox networks that are useful in multiple, general ways for development and design-- the analogue of metaphor is genetic pleiotropy, in which genetic structures are reused for multiple computations. Once such instructions and modules have been discovered, evolution is no longer a random syntactic search, but rather is searching over meaningful possibilities. It took evolution billions of years to discover Hox networks, but since then has been experimenting by swapping around meaningful combinations (bigger brain, longer legs, more body segments), expressible by program changes as short as adding a single instruction, and has built us in only another half billion years.

Evolution has evolved to evolve better. This is again controversial, what Gilbert et al (1997) call it "a wrong teleological view" that mechanisms may be selected for because "they aid future selection". But evolution discovers a compact program that has solved numerous learning problems, and the Occam thesis says: a compact program that has solved numerous learning problems in the past will be good at solving new ones it has not yet encountered.

Much recent heat has been generated by arguments that neo-darwinism can not explain the origin of as much structure as we see in living things, thus it is claimed that a teleological explanation is needed: intelligent design. The answer is not posit an omnipotent God (which, to the extent that it could explain anything at all, can not be indicated in an Occam sense by any particular data whatsoever). It is to understand the origin of intelligent design. Intelligent design as practiced by humans, I believe, involves a search over alternatives in which the alternatives are only those judged relevant by an extensive existing computational machinery, and involve the combination of large, meaningful computational modules to specify. This human ability was produced by evolution, which also evolved to apply the same general technique in its own advancement, and evolved so for very similar reasons.

Nature v. Nurture

The model as so far explained asserts that the genome encodes programs causing development, or near-automatic learning, of meaningful computational modules, and that thought is execution of an Occam program built on such modules.

Given the existence of such a modular program, there is a natural model for language. Words are labels for meaningful computational modules. Attaching such labels allows communication of programs.

Since many of these modules and much of this program, if encoded in the genome, must be present already in animals, this raises the question of why language took so long to evolve. A plausible answer is that evolution was stuck in a state from which it had to make too large a leap. One such trap, proposed by Martin Nowak and collaborators, was analog encoding. Until one has a vocabulary of about 20 words, it turns out to be fitter to use analog encoding, a fixed word for each concept. In this picture, it took evolution a long time to make the jump to digital encoding, in which words are formed of syllables and sentences of words. One can't readily evolve digital encoding till one uses more than 20 words, and one can't readily evolve to use more than 20 words until one uses digital encoding. Animals like vervet monkeys do indeed use analog languages of less than 20 words. Another such trap is of discovering the idea of names. If I am naming concepts, you might easily evolve to learn the names. If you are learning names, I may easily evolve to name concepts. But evolution can not progress to language until naming and learning names are simultaneously discovered, which may take some time. It may indeed have been first accomplished by a single pair of cave-person Einsteins.

The ability to communicate programs, once found, is extraordinarily powerful, in particular because it allows sustained development of new programs.

The human mind evidently employs algorithms not explicitly coded in the genome. For example, the reader has procedures that allow him or her to read this text. Such knowledge is built as meaningful modules that invoke more basic modules. Computer science argues that complex programs must be built in such a modular fashion, sometimes called an abstraction hierarchy. The Occam's razor hypothesis suggests that the modules coded in the genome are meaningful precisely in the sense that powerful programs can be built on top of them. That is: these compact modules are such that in past experience, powerful programs have emerged to solve problems such as navigating in the jungle, therefore these modules should be such that powerful superstructures will emerge in new domains such as reasoning about higher mathematics.

Nonetheless, complexity theory suggests that finding new meaningful modules is a hard computational problem, requiring substantial search. This suggests a mechanism by which human mental abilities differ so broadly from chimpanzees, who are genetically almost identical. Chimpanzees can discover new meaningful modules only over one lifetime of study. But humans, because of our ability to transmit programs through language, have cumulatively built and refined a vast library of powerful computational modules on top of what is coded in the genome.

This additional programming does not just include what we think of a technology, but rather includes qualities regarded as core mental abilities. For example, only humans are said to have a theory of mind in that we understand that other individuals may have different beliefs than our own, and reason accordingly. However, apes display behaviors indicating aspects of theory of mind; and plovers feign injury, limping off to distract a predator from their nest, only when the predator seems to notice the nest. Thus plovers attend to the perception of the predator and modify their behavior accordingly. This ability requires subroutines on which any theory of mind must rely. This suggests that the human theory of mind is a complex modular program built on top of meaningful modules coded more explicitly in the genome, and that humans have been able to discover this more powerful program over generations, because we pass partial progress on. Bedtime stories and fiction are means of passing such programs on to children. Psychophysics experiments are consistent with this view, showing that children acquire theory of mind over a period of years.

Thus this computational theory of mind, while it suggests that much of thought is coded in the genome, simultaneously suggests that most of the difference between human and ape cognition can be regarded as the product of better nurture. The key is that the program of mind is modular and hierarchic, and that the discovery of new modules is a hard computational problem. By virtue of our ability to communicate partial results, humans have made sustained progress in developing an ever more powerful superstructure on top of the innately coded programming.

The Evolution of Consciousness

The hypothesis that every thought is execution of evolved, meaningful code also models all the many aspects of consciousness, such as will, awareness, and our sense of experience. Evolution produces a program that chooses actions based on sensory data in a way that promotes propagation of the genes. Such a computation can be regarded as a two part system: a decision making unit and an internal reward function. The decision making unit makes decisions based on current information to maximize its long term internal reward. The internal reward function is programmed by the genome to guide the decisions to maximize propagation of the genes. The internal reward function is detailed, including gradations of pain and pleasure and satiety and not wanting one's kids to cry. Emotional states are another manifestation of the genome exerting control over the decision making process.

We think about the world using Occam modules: concise programs consistent with data that do the right thing. We live in a world of creatures that are computer programs attempting to maximize internally computed goals. The behavior of such is intrinsically hard to predict. Since we can not estimate exactly how creatures will behave, we have evolved Occam modules that model it simply. What we mean when we think something has free will is that we reason about its behavior using code that calls such an Occam module, and we naturally apply the same reasoning to predict our own future actions.

Philosophers such as Searle accept that the physics of the brain is deterministic, or at most randomized by quantum effects. They argue however that what they mean by "free will" is not simple randomization, that the supposed existence of a genuine "free will" implies that "some process we don't understand" injects genuine "new information" at the moment a choice is made. They argue this is inconsistent with a Turing machine picture. However, Turing proved that the outcome of computer programs, although they are reducible to simple rules, are inherently unpredictable by any method better than simulating the program exactly, which can take arbitrarily long. Thus the decisions made by a Turing machine, although each intermediate step of the calculation is transparent,

are overall as inscrutable as any philosopher could desire. In principle our choices could be predicted by simulating exactly the physics of the brain on a computer. But new information appears for all practical purposes, since such prediction is impossible.
Our understanding is based on an evolved program that reasons about the world in a practical manner.

The mind's decision making program is a hierarchic modular structure. It contains many submodules that extract from raw sensory data meaningful information. At or near the top of this hierarchy is a module (or a collection of modules) that we may name the homunculus that calls such submodules and the modules computing internal reward, and makes decisions, passing these onto driving utilities for muscles.

You are not aware of all the processing that extracts meaning from raw sensory data. For example, the color red that you perceive is not simply related to the frequency of light impacting your retina, but rather is the output of complex modules processing raw sensory input to estimate an intrinsic attribute of the object that is independent of its illumination. A strawberry will thus look red whether you see it at noon, or under a blue lamp, and you feel nothing of the processing that produces this sensation.

Nor can you report the internal computations of the homunculus. Decisions appear. Words appear. The intermediate values within a meaningful computational module are meaningless, and the homunculus is thus no more programmed to report them than a laptop can report the voltages at its internal transistors.

However, our sensation of awareness is identically the homunculus's ability to report inputs directly weighed in its decision. Chalmers has asked whether there might in principle exist a zombie that reacts as you do but without sensation. The answer is no, because it is precisely the act of qualitative decision by the homunculus that should be identified with sensation. Pain feels intrinsically awful precisely because the homunculus has been programmed by evolution to regard it as awful. A rose smells sweet, because the rose has evolved to exploit preexisting coding within the brain in exactly this way, so as to appeal to honeybees. Thus the Alaskan Lily smells like fecal matter because it has evolved so to appeal to flies, which pollinate it.

Many amputees feel themselves clenching their phantom hand so hard the phantom nails dig painfully into their phantom palm. Vijay Ramanchandran predicted that this occurred when the patient's brain sent a command to relax his or her hand, but receiving no confirmatory feedback, assumed the hand continued clenching. Ramanchandran built a mirror device into which patients could insert both phantom and real hands, unclench both simultaneously, and receive visual feedback (observing the real hand in the mirror) that both had unclenched. Instantly, the pain vanished.

Once one makes the ansatz that all thought is execution of evolved, meaningful computer code, one gets a parsimonious model that naturally explains all aspects of consciousness, and is consistent with all data of which I'm aware, subjective and objective. Psychophysics experiments such as Ramanchandran's will continue testing this theory. Likewise it will be tested by progress in understanding the genetic circuitry both for sensations such as itchiness or joy and for other computational modules. Finally it will be tested by brain imaging, which should increasingly reveal the modular structure of thought and the way reuse of modules corresponds to metaphor.

Will humans ever create machines that understand and are conscious? The Occam theory implies it is possible, but difficult. Improving technology has put us in reach of computers with the raw computing capabilities of the human brain, but no technology in sight will provide computers to rival the computing power biological evolution applied. It was this massive power that produced the software of thought. Unless scientists can find computational shortcuts, thinking machines may remain beyond our reach.

Transcendental Morality

Some anti-physicalists worry that absent a God, or at least absent something more to consciousness than an evolved program, there can be no definition of morality in the world. Some physicalists agree, arguing there is no reason to believe morality exists in the sense of inhering in the universe as an objective property. However, it is possible that morality will be derived from mathematics, and is thus as fixed, absolute, and discoverable as the ultimate laws of physics. In particular, it may be that there is a well-defined strategy for multiple parties to interact that is optimal in a natural sense.

There have been attempts to derive moral principles within game theory, but it has not yet proved possible to treat rigorously two important factors, coalition forming and progress.

The evolution of economic progress, as exemplified in our experiments in evolving artificial economies of computational agents, serves as a motivating example. These programs interacted with a simulated world, and progress was possible as knowledge of how to exploit the physics of that world was discovered and applied. Under such circumstances, all parties can improve their lots over time if they can enforce cooperation. Our experiments as well as several arguments (detailed in What is Thought?) indicated that a very simple, concise, natural law of property rights can lead to rapid progress, that violations of such natural law can lead to undesirable consequences, and moreover that evolution discovered methods (such as posses among humans and suppression of cheaters among genes in meiosis) to enforce such natural law.

The point may be more general. Arguably there is a symmetry-breaking arrow to evolution: over time it is possible to learn better how to exploit the compact structure of the world. Moreover, cooperation furthers such ``progress", and simple rules apparently exist that promote cooperation. But this renders such rules evolutionarily fitter, and because they are simple, and thus evolvable, they evolve. Evolutionary history has seen the evolution of cooperation at multiple levels, such as the cooperation of genes in chromosomes, the cooperation of cells in bodies, the cooperation of individual creatures in a societies. But quite plausibly the evolved rules promoting progress and cooperation are intrinsic, as well as very simple. One might find very similar solutions if one were able to run the tape again on another planet.

E. O. Wilson wrote ``Centuries of debate on the origin of ethics comes down to this: Either ethical precepts, such as justice and human rights, are independent of human experience or else they are human inventions." The former view he calls transcendentalism, the latter empiricism. Transcendentalists hold that ethical precepts are, in the words of the Declaration of Independence, ``self-evident... endowed by [the] Creator". Wilson argues for empiricism, saying that morality follows from epigenetic rules: "ethical codes have arisen by evolution through the interplay of biology and culture". And he claims ``is there a way to ... resolve the contradictions between the transcendentalist and empiricist world views? No, unfortunately there is not."

The two views can however be resolved. The conjecture that there are absolute ethical precepts arising from the nature of the universe is clearly transcendentalist. But such absolute principles would then be reflected in the program of mind just as the nature of physics is reflected in the program of mind. As Wilson argues, evolution has extracted epigenetic rules predisposing human behavior and thought. But if my conjecture is correct, these rules reflect the underlying transcendental nature of the universe.

People's understanding of morality then derives from this conjectured absolute morality, but should not be expected to exactly mirror it for various reasons. The intuitive understanding of morality produced by evolution should not be expected to be a more precise picture of absolute truth, the underlying compact structure of the universe, than was the intuitive understanding of physics. But intuitive physics, of course, mistakes Newton's, Galileo's, Copernicus's and even Archimedes's discoveries. More importantly, evolution may have produced systematic divergences from absolute morality. For example, it may well have been fitter in Darwinian terms to adapt a submissive posture if one's leader is bigger and stronger, as do monkeys, chickens, and dogs. However, it is plausible that absolute morality is more egalitarian. Most importantly, cultural programming seems to have run away with notions of morality. Because cultural programming has worked so rapidly, evolution has not had time to push it toward optimality, and thus a profusion of cultural programming is superimposed on any existing epigenetic rules.

Wigner once marveled at ``The unreasonable effectiveness of mathematics in the natural sciences." We live in a world that has a beautiful underlying simplicity. It may be that this simplicity extends into morality as well, and thus that there is a simple natural law of ethics just as there are simple principles underlying physical interactions. Like the laws of physics, this natural law of ethics may yield to mathematical and scientific investigations.

Morality, as well as thought, physics, and mathematics may stem intrinsically from a world that is near optimal in the sense that it possesses simultaneously the simplest possible underlying structure and the richest possible phenomena.