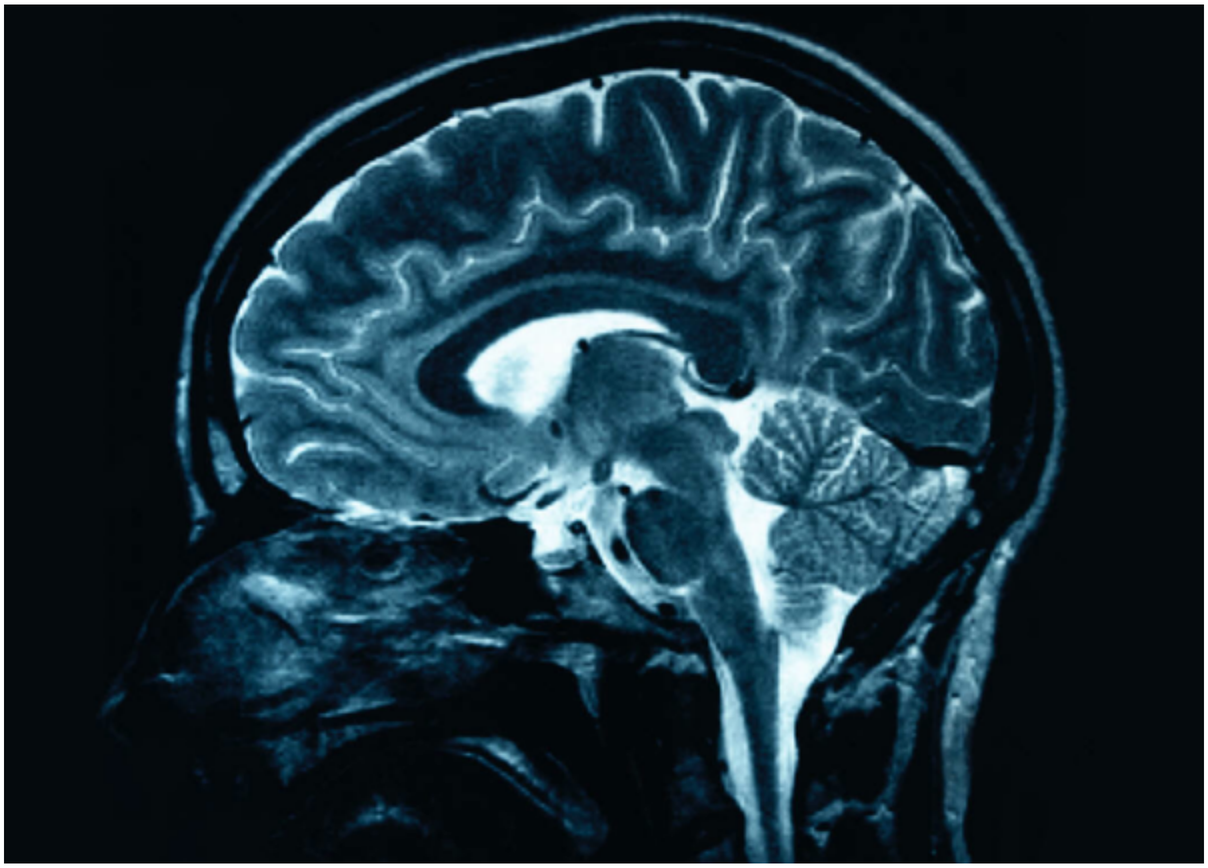
Cognitive Science in One Lesson

by Luke Muehlhauser on January 10, 2011 in Resources,Science



It is always useful to know something about the tools you use. We require that you take driving lessons before you drive, and you won't be a very good photographer if you don't understand how a camera processes light.

What tools do you use when you think philosophically about God or morality or other subjects? Among other tools, you use *your mind*. Knowing how the mind works can help you do philosophy better, just as knowing how a camera works can make you a better photographer.

Cognitive science is the science of the mind, and its results are the most we know about the mind so far. To help us all do philosophy better, I'd like to summarize what we know so far about our mind-tool. I will do it by summarizing an excellent introductory textbook on cognitive science by Jose Luis Bermudez, director of the Philosophy-Neuroscience-Psychology program at Washington University in St. Louis.

If you want more details on *any* of what is below, I highly recommend you buy the book.

Contents:

As you will see, cognitive science is a highly interdisciplinary field of study, drawing on results from psychology, neuroscience, mathematics, computer science, philosophy, linguistics, molecular biology, and other fields.

The guiding idea of cognitive science today is that **mental processes involve the processing of information**.

But this was not always the case. How did we get there? Bermudez covers 4 developments that led us here:

- The reaction against behaviorism in psychology
- The idea of algorithmic computation in mathematical logic
- The emergence of linguistics
- The emergence of information-processing models in psychology

Against Behaviorism

Behaviorism dominated the field of psychology for decades. Its basic idea was that psychologists must confine themselves to studying measurable behavior, and avoid speculating about unobservable mental states like "belief" and "desire" and "thought." Since cognitive science is largely the science of mental states, cognitive science could not begin until behaviorism declined.

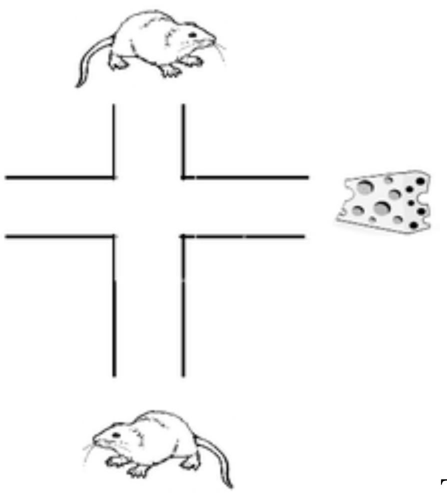
Why did behaviorism decline? One key study was done by behaviorist psychologist Edward Tolman in 1930. Tolman had standard behaviorist assumptions about learning: he believed that learning is the result of conditioning that results from association and reinforcement.

To illustrate, consider a rat in a box. Each time the rat presses a button, it receives a food reward. The reward reinforces the button-pressing behavior, because the behavior and the reward are associated again and again, and so the rate becomes conditioned to press the button.

The 1930 experiment studied how rats learned to run mazes. Each rat from the first group of rats received a reward upon completing the maze. The second group of rats never received a reward. The rats in the third group went unrewarded for 10 days and *then* began to receive rewards upon completing the maze. As behaviorism predicted, the rewarded rats learned the maze quickly, and the two groups of unrewarded rats wandered aimlessly.

But when the third group began to receive rewards upon completing the maze, it learned the maze even more quickly than the first group had. Tolman argued that the rats in the third group must have been learning the layout of the maze even when they were not being rewarded, and that's why they figured it out so quickly once they *were* being rewarded. The rats had been learning even while not being rewarded, so reinforcement was *not* necessary for learning.

One follow-up question was: When the rats were learning the maze, what kind of information were they storing? Were they simply "remembering" the sequence of movements (left turn, left turn, right turn) that got them from the start of the maze to the food reward? Or were they building a mental map of the maze in their heads so they could navigate it?



Tolman and others tested this by building a simple cross maze. Rats were allowed to run the maze many times, but their starting point was changed back and forth between the north and south entrances.

For the first group of rats, the food reward was always placed at a fixed location: say, the east exit. For this group of rats, the same turning behavior would not always result in reward. When starting from the south, these rats would have to turn right to get the reward. When starting from the north, these rats would have to turn left to get the reward.

For the second group of rats, the food reward was moved between its east and west locations so that the same turning action would always result in a reward. In this group, a rat starting from the south would find its reward to the east (a right turn), and a rat starting from the north would find its reward to the *west* (also a right turn, coming from the north).

To reach the reward quickly, the second group of rats would merely have to remember a particular sequence of movements (a right turn). But for the first group of rats to reach the reward quickly, it would have to make a cognitive map of the maze and represent its place in it and the food's place in it.

Surprisingly, Tolman found that the *fi rst* group of rats learned the maze more quickly. Tolman concluded that a rat learned a maze not by having certain sequences of movements programmed into it by reinforcement, but by representing a model of the maze in its mind. This concept of *mental representations* is now one of the key ideas in cognitive science.

In 1951, Karl Lashley published a paper on how to explain complex behavior. He noted that when we engage in a complex series of movement – for example speaking a sentence or playing a game of tennis – it is *not* the case what happens is continuously prompted by what is going on in the environment. Instead, these actions are greatly determined by the *goal* of the behavior. Each plan (say, to hit the tennis ball over the net) is implemented by many simpler plans (step backward and to the right, swing the racket, twist the wrist just as the racket hits the ball, etc.).

Lashley's major idea is that complex behavior is not a *linear* sequence of responses to reinforcers in the environment, but rather the result of planning and information-processing that is *heirarchical* (big plans broken into smaller tasks) and often *unconscious* (you usually don't *consciously* calculate the rotation of your wrist when hitting a tennis ball). This "hypothesis of subconscious information processing" and "hypothesis of (hierarchical) task analysis" remain two of the key ideas of cognitive science today.

Computation

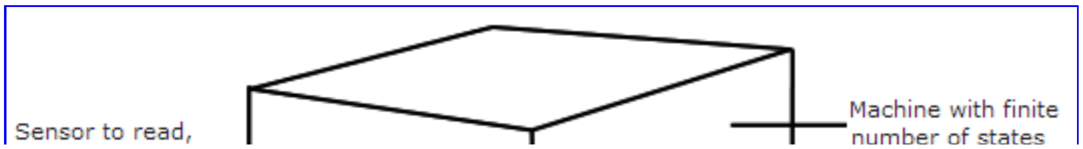
In 1937, Alan Turing described the notion of a Turing machine, which formalized the modern concept of computation, which is what computers do and, according to many cognitive scientists, what minds do.

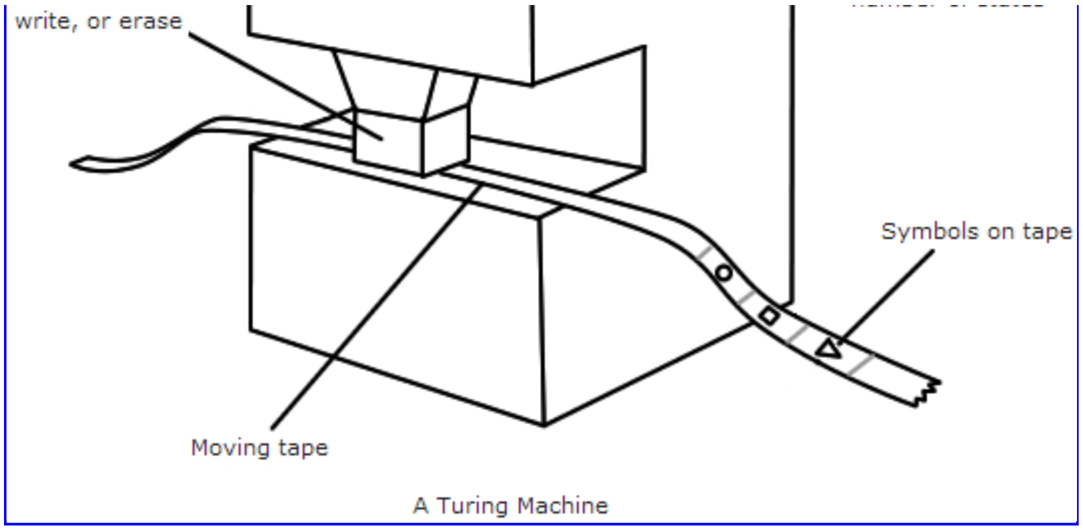
What is a Turing machine? Imagine an infinitely long piece of tape divided into cells like a film strip. Each cell can be either blank or filled with a single symbol. The tape runs through a machine head, with a single cell under the machine head at any given moment.

The machine head reads the cell, and can perform one of the following acts after reading it:

- It can delete the symbol in the cell.
- It can write a new symbol in the cell.
- It can move the tape one cell to the left.
- It can move the tape one cell to the right.

The machine can also be in a certain number of internal states. What the machine head does is determined by a set of instructions called its *machine table*. The machine table determines what the machine will do when it encounters a particular symbol in a cell and is in a particular internal state. The machine has no intuition or judgment – it mechanically does what is determined by the machine table, its current state, and the symbol in the cell it is currently scanning.





Turing did not build this machine, but he specified it mathematically, and was able to prove there is a special kind of Turing machine, a Universal Turing machine, that can run *any* specialized Turing machine. And this, in fact, is exactly what your laptop is. Your laptop can perform any kind of computation (within the limits of its memory size). That is why your laptop can surf the web, play music, play games, edit video, analyze statistics, or convert text into spoken words.

Turing's model of computation is now a popular model for how the mind processes information. If the mind processes information *computationally* – rather than by intuition or judgment coming from beyond the mechanical processes of the brain – then there is no need to suppose a homunculus or a "ghost in the machine." And in fact, Bermudez writes that "Few, if any, cognitive scientists are *dualists*, who think that the mind and the brain are two separate and distinct things." The mind just *is* the brain, and appears to process information mechanically in something like the way that Turing modeled.

The structure of language

Noam Chomsky's 1957 book *Syntactic Structures* was perhaps the first attempt to explain *why* languages work as they do, rather than just describing *how* languages work. Chomsky proposed that below the *surface structure* of a sentence (the actual layout of words in a sentence), there was a *deep structure* of how it is built up from basic elements (nouns, verbs, etc.) according to basic rules (for example, every grammatical sentence is composed of a verb phrase and a noun phrase).

Consider two sentences: "John has hit the ball" and "The ball has been hit by John." Though their surface structure is different, Chomsky explained that their deep structure shares important features that can be illuminated with *transformational grammar*.

The principles of transformational grammar are mechanical *algorithms* (like the rules in the machine table of a Turing machine) that convert one string of symbols to another. Consider the following algorithm:

> NounPhrase1 + Auxiliary + Verb + NounPhrase2 → NounPhrase2 + Auxilery + "been" + Verb + "by" + NounPhrase1

This algorithm transforms “John has hit the ball” into “The ball has been hit by John.” It also transforms “The mechanic has fixed the car” into “The car has been fixed by the mechanic.” And so on. It is a mechanical rule that shows the *deep* structure similarity between two sentences of very different *surface* structures.

Transformational grammar brings together Lashley’s idea that complex cognitive tasks can be organized hierarchically (in this case, a sentence is broken down into smaller elements) with Turing’s idea that information processing tasks can be performed with mechanical algorithms.

Information-processing models in psychology

Bermudez illustrates the arrival of information-processing models in psychology through two famous papers of the 1950s.

In 1956, George Miller surveyed a wide range of evidence and argued that our senses use information channels (paths between senders and receivers of information) that can only transmit a certain amount of information at a time. In particular, he proposed that our senses can only transmit about “seven, plus or minus two” pieces of information at a time.

For example, consider one experiment in which subjects were asked to assign numbers to the pitches of certain sounds. They were then played a sequence of sounds and asked to give the corresponding sequence of numbers. Let’s say middle C = “1″, the first E above middle C = “2″, and the first F# above middle C = “3″. If the subject heard C, E, F#, E, E, then the correct respond would be 1, 2, 3, 2, 2.

Subjects almost never give the wrong response when the sequence is one or two sounds long, but their performance falls off drastically when sequences are about 7 sounds long. The same thing happens when we ask people to remember a sequence of differently sized squares or lines of different lengths. Whether working with hearing or seeing, we seem to only be able to process about 7 distinct pieces of information at a time.

One way around this limitation is to “chunk” information, which is what you do when you try to remember a telephone number. You do not remember “800-555-0867″ as “eight-zero-zero-five-five-five-zero-eight-six-seven.” That would be too hard. Instead, you probably remember it as “EightHundred-ThreeFives-zero-eight-six-seven”.

The psychologist Donald Broadbent built on this idea and surveyed a large number of experiments to form his model of selective attention:

> Information comes through the senses and passes through a short-term store before passing through a selective filter. The selective filter screens out a large portion of the incoming information, selecting some of it for further processing. This is what allows us selectively to attend to only a portion of what is going on around us in the cocktail party. Only information that makes it through the selective filter is semantically interpreted, for example. Although people at cocktail parties can hear many different conversations at the same time, many experiments have shown that they have little idea what is said in the

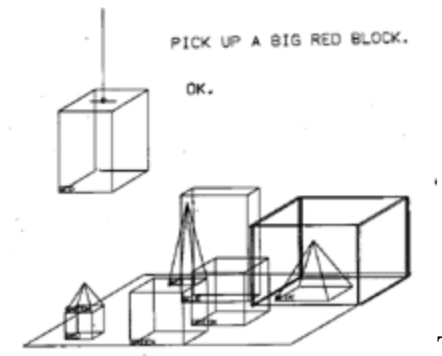
conversations that they are not attending to. They hear the words, but do not extract their meaning.

Reproducing natural language with algorithms

These studies give us some clues about how the mind could cause complex behavior like speaking and playing tenns by using mechanical algorithms to process information. But the algorithms themselves are still unknown. One way to get at which algorithms the mind might be using is to reproduce human thought with computer-coded algorithms.

And that's just what Joseph Weizenbaum did in 1966 with a program called ELIZA. ELIZA contained a series of rules for how to respond to sentences typed by a human. You can have a conversation with ELIZA here. A more sophisticated "chatterbot" from 2008 is Elbot.

In 1970, Terry Winograd wrote a program that could not only simulate a conversation, but also report on its environment in natural language, plan actions, and reason about the implications of what was said to it.



The program, SHRDLU, lives in a micro-world consisting only of a few colored blocks, colored pyramids, and a box. SHRDLU can move objects around with its virtual robot arm.

Let's say we told SHRDLU to "Find a block which is taller than the on you are holding and put it in the box."

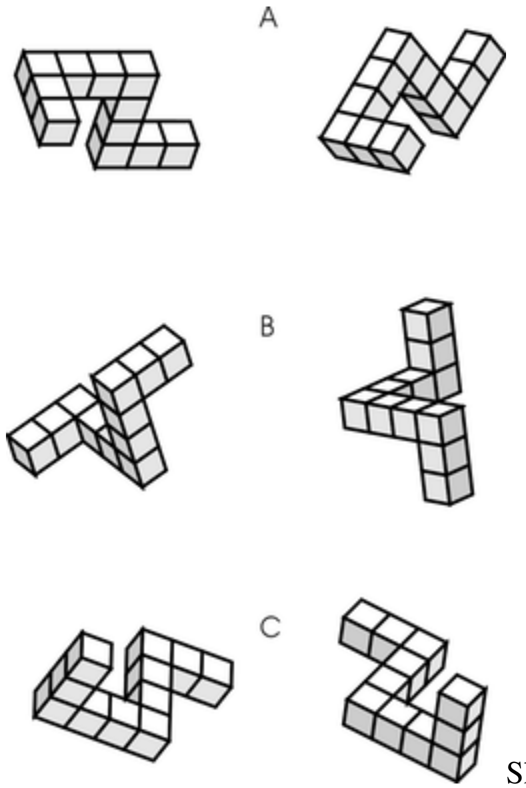
First, SHRDLU would decode the grammar of the sentence: it would decode its nouns and verbs. Then, it would determine the meaning of the words themselves. It would then apply this information to information it already has about, for example, which block it is currently holding, the height of the blocks in its micro-world, and the location of each block relative to the box. It would then carry out the command by dropping the current block, grabbing a taller block, and moving it into the box.

Of course, each of these tasks is broken into a large number of smaller tasks. For example, the procedure for decoding the grammar of the sentence includes algorithms for recognizing whether a verb is transitive or intransitive, and how to treat the rest of the sentence as a result of that answer.

SHRDLU was important in the development of cognitive science for three reasons. First, it illustrated how abstract rules such as those of human language could be implemented *algorithmically*. Second, it was a powerful example of how we might model cognitive systems by breaking them down into smaller and smaller information-processing tasks. Third, it illustrated the importance of cross-talk between the systems that implemented all these different mini-tasks.

Imagistic representation

How does the brain encode and store information? Perhaps it stores information as a series of bits, a series of 1s and 0s, like a computer does. One famous experiment that sheds some light on this question is Shepard and Metzler's 1971 study of mental rotation.



Shepard and Metzler presented subjects with pairs of 3D objects and asked them to mark the ones that were identical. They found that the time it took subjects to identify identical 3D objects was directly correlated with the degree that one object was rotated from another. It appeared that subjects were *not* subconsciously running an algorithm on a table of bits of data, but rather they were mentally rotating image-like representations in their head to see if two 3D objects were identical. Indeed, that is what many subjects *reported* that they had done to compare each pair of objects.

In another study, Stephen Kosslyn asked subjects to memorize a series of drawings, for example of a lighthouse and an airplane and a boat. He took the drawings away and asked subjects to focus their memory on one end of one of the objects (e.g. "the tail of the airplane"). He then asked subjects to recall whether the object had a certain part (e.g. "propeller").

He found that the time it took subjects to answer was correlated with the distance of the named part from the point of focus. If the subjects were asked to focus on the tail of the plane, it would take them longer to confirm that the drawing of the plane had included a propeller than that there was not a pilot in the cockpit. Kosslyn concluded that answering these questions involved mentally scanning imagistic representations, not searching a digitally encoded database of information about the drawings.

Thus it seemed that at least *some* of the information that our minds process is encoded as images rather than as digital bits like in a computer.

David Marr on vision

David Marr (1982) distinguished three levels of explanation in cognitive science. The first is the *computational level*, at which we try to identify the information *input* into the system (say, from the senses) and its *output* (say, behavior). What is the logic of how this happens? These are our tasks for understanding the mind at the computational level.

The next level down is the *algorithmic level*. Here, we try to identify which specific algorithms are used by the brain to achieve the computational goals discovered at the level above. For example: By which algorithm does the mind transform information from the visual system into a mental representation of a 3D shape?

The third level is the *implementational level*. At this level, we try to understand how the algorithm from the above level is realized in the physical structures of the brain.

Marr then applied his three-level system of analysis to the visual system. At the computational level, he drew heavily from research on brain-damaged patients, in particular on work by Elizabeth Warrington. In one study, Warrington noticed that patients with lesions on their right parietal lobe are able to identify familiar objects if they are seen from a conventional angle, but not if they are seen from an unusual angle. In contrast, patients with lesons on their *left* parietal lobe could identify familiar objects from conventional *and* unusual angles without difficulty.

Marr concluded that information about the *shape* of an object must be processed separately from information about *what the shape is called*, and that the visual system can identify shape without knowing what the shape *is*. So at the computational level, it appears that the visual system is designed to first derive the spatial arrangement of an objects so that it can later be recognized for what it is.

At the algorithmic level, then, how is this kind of computation achieved? Marr proposed that a mental representation of an object was arrived at by way of a series of increasingly complex representations, which he called the *primal sketch*, the *2.5D sketch*, and the *3D sketch*.

The primal sketch includes basic information about areas of lightness and darkness from the retinal image. The next step extracts from the primal sketch information about the depth and orientation of the *surfaces* in the field of view, and their distances from the viewer. The third step derives a 3D sketch of the object from the 2.5D sketch. This is what allows a person to recognize the object later from a different angle. The 2.5D sketch is viewer-centered, but the 3D sketch is object-centered, and that is why we can recognize the object later from a different angle, and why we can tell that two 3D shapes are the same shape, rotated.

Marr went into much more detail than this on the algorithmic level, but said little about the implementational level, in part because the neurochemistry and molecular biology of the brain was not as well understood as it is today.

Two visual systems

Many cognitive scientists argue that cognitive systems should be studied as *functional* systems that serve certain functions regardless of the physical matter on which they run. For example, the function of a *heart* is to pump blood around the body, but whether a heart has four chambers (as in humans) or three chambers (as in most reptiles), or whether it is an *artificial* heart, is not so important. Functional systems, most believe, are *multiply realizable*. They can be physically realized in a number of ways, as long as they serve the same function.

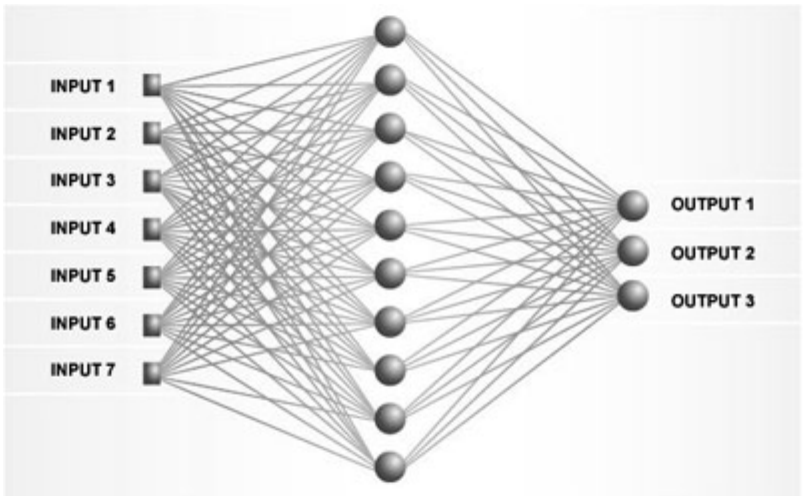
This may be true, but it has still been enormously productive to study the physical brain and how *it* realizes cognitive systems. One study that illustrates the "turn to the brain" of 1980s cognitive science was conducted by Leslie Ungerleider and Mortimer Mishkin 1982. Their study drew on brain damage cases and from neuroanatomical experiments in monkeys, and concluded that visual information takes not one but *two* routes from the visual cortex at the back of the brain. Information relevant to *recognizing* objects takes a ventral (bottom-of-the-brain) route to the temporal lobe, and information relevant to locating objects in space takes a dorsal (top-of-the-brain) route to the posterior parietal lobe.

Their study was an important step in mapping the connections in the brain and mapping the flow of information within it. It is also an example of a cognitive task being broken into smaller and smaller tasks for processing by different parts of the brain that perform different sub-functions.

Neural networks

At the same time, more abstract attempts to mathematically model cognitive processes were also productive. A 1986 collection of papers by David Rumelhart and others proposed the concept of a *neural network* (or *connectionist network*), a computational model of information-processing designed to emulate the connections of neurons in the brain.

One key feature of a neural network is *parallel processing*. To get a picture of how a neural network works, think of it as containing a large number of *units* (artificial neurons, basically) which are organized into *layers*. Each unit in one layer has connections running *to* it from units in the *previous* layer, and also has connections running *from* it to connections in the *next* layer.



Each unit has a *level of activation*, given as a real number between -1 and +1. The level of activation for an entire *layer* is determined by the level of activation for each unit in that layer. Information flows through the network according to the activation level of each layer, which is determined by the level of activation of all the units in that layer, even though the units in each layer are not connected to each other. This is why we call the processing "parallel."

Moreover, the strength of a connection between two units has a certain *weight*, which is can change over time. But the units themselves are interchangeable: what defines the network is the pattern of weights for the connections *between* the units.

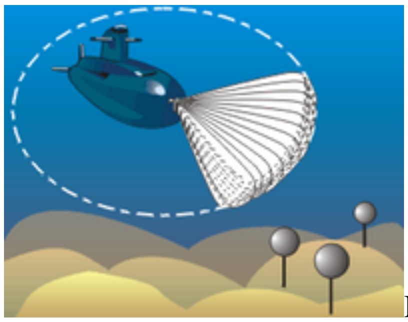
Another interesting feature of a neural network is that the weights of each connection usually aren't programmed explicitly. Rather, a broad plan is programmed, which continuously modifies the weights of the connections so that the network gradually makes fewer and fewer "mistakes" in achieving the goals with which it has been programmed.

In the diagram of a neural network above, the left-most layer receives input from outside the network. The third layer sends output signals outside the network. The middle layer is made of *hidden units* – "hidden" because they have no direct contact with anything outside the network. Of course, real neural networks almost always have *many* layers of hidden units, so that more complex processing can be achieved.

Training a neural network can take some time. Often, we begin with a random assignment of connection weights and then give the network a series of input patterns, each of which has a target output pattern. Differences between the actual output and the actual output pattern results in changes to the connection weights. This is how the neural network "learns" – trial after trial, it adjusts the weights of its connections to reduce the difference between the actual and desired output.

This process, the "backpropagation of error," continues until the error between actual and desired output has been diminished to very nearly zero.

But this is all rather abstract. Let's look at a real example: a neural network designed to detect whether a particular underwater sonar echo comes from an explosive mine or a rock.


Human sonar operators, after much training, can do pretty well at detecting mines from sonar data, but neural networks can now do much better.

One such neural network was designed by Paul Gorman and Terrence Sejnowski. It works like this: First, sonar echoes are translated into a pattern of activation levels so that each sonar echo has a unique "fingerprint" that can serve as the *input* for the neural network. The input layer has 60 units unique "fingerprint" that can serve as the *input* for the neural network. The input layer has 60 units. There is one layer of hidden units, and the output layer has just two output units: one for classifying a signal as indicative of a rock, and the other for classifying a signal as indicative of a mine.

The network is a "feedforward" network, which means activation only ever spreads *forward* through the network, not backward. During the training process, the network receives input from sonar echoes that are known (by the designers) to be indicative either of a rock or a mine. At first, the activation through the network spreads randomly, because the weights of the connections have been assigned randomly. But on each trial, whenever the network comes up with the wrong output pattern of activation (say, <0.75, 0.1> instead of <1, 0>), the error is backpropagated through the network and the connection weights are adjusted to reduce the error. After many, many training runs like this, the network can accurately detect mines with 90% accuracy.

The most common tasks for neural network is pattern recognition of this kind. We now have neural networks that can detect faces, unwanted email, handwriting, medical conditions, and much more.

The idea of a neural network can help us think about how the brain processes information, and other tools now allow us to actually *watch* the flow of information in the brain. (We'll cover backpropagation of error and neural networks and so on in more detail, later.)

Brain scanning

One of the early brain-scanning methods was *positron emission tomography* (PET), which tracks the flow of blood in the brain. We can see which parts of the brain are being used by certain cognitive tasks by identifying the areas to which blood is flowing while those tasks are being carried out.

In PET, subjects are injected with a tiny amount of radioactive water, which accumulates in the brain in direct proportion to the blood flow in that area. Of course, one of the challenges of PET experiments is to distinguish blood flow associated with background tasks and blood flow associated with the cognitive task we want to investigate.

One classic PET study was conducted by Steve Petersen and his colleagues in 1988. Petersen wanted to understand how brains process individual words.

There are two leading theories of single-word processing. The *neurological* model claims that processing a word is *serial*: information flows through a series of "stations" in a fixed order, each of which performs a different task. One station processes auditory information about the word, another processes its visual appearance, another processes its semantic content, and so on.

The *cognitive* model of word processing is *parallel*. It claims that different types of processing is performed on a word at the same time, and they can feed each other through multiple channels.

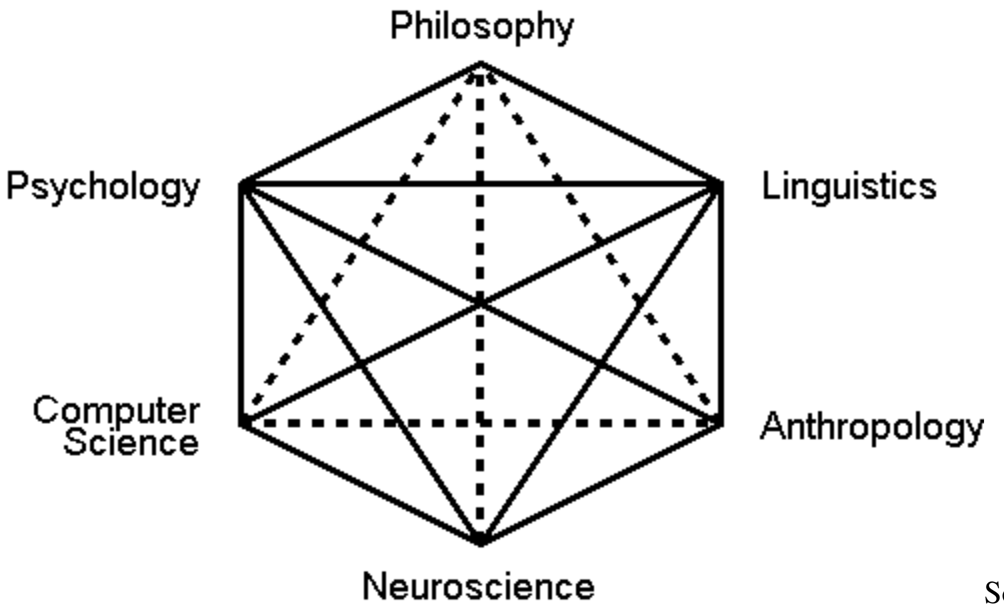
To test between these two models, Petersen set up an experiment with several conditions. In the first condition, subjects' brains were scanned while they stared at a small cross-hair in the middle of a TV screen. This showed what parts of the brain were active when looking at something that is not a word. In the second condition, subjects were shown words on the screen at 40 words per minute. In the third condition, these words were *spoken* to the subjects.

To see what brain areas are involved in *visually* processing a word, Petersen subtracted the first-condition scan images from the second-condition scan images. And to see what brain areas are involved in the *auditory* processing of a word, Petersen subtracted the first-condition scan images from the *third*-condition scan images.

Petersen repeated the process to find the brain areas associated with *speaking* the given words, and also for speaking a verb the subject thought of when seeing the given noun.

After statistically analyzing all the images, Petersen found that these four tasks involved very different sets of brain areas. Also, activation did not proceed *serially* through a fixed series of brain areas, as in the neurological model. For example, simply viewing the words did not activate the brain areas associated with *listening* to the words at all.

So far we have seen that (1) the mind is the brain, which is an information-processing system, and (2) cognitive science integrates a huge variety of fields, as illustrated by this famous 1978 diagram:



Solid lines depict strong interdisciplinary links, while broken lines depict weaker interdisciplinary links. Bermudez writes:

> Each of the six disciplines brings with it different techniques, tools, and frameworks for thinking about the mind. Each of them studies the mind from different perspectives and at different levels. Whereas linguists, for example, develop abstract models of… [the] structure of language, psychologists of language are interested in the mechanisms that make possible the *performance* of language users. Whereas neuroscientists study the details of how the brain works, computer scientists abstract away from those details to explore computer models and simulations of human cognitive abilities. Anthropologists are interested in… how cognition varies across cultures. Philosophers, in contrast, are typically interested in very abstract models of how the mind is realized by the brain.

Of course, things have changed since 1978. Even more fields are now involved, and some of the links above have been stronger. For example, the link between neuroscience and philosophy would now be considered "strong," and there is a whole group of researchers who call themselves, simply, neurophilosophers.

This brings us to the integration challenge: how can we develop a unified model for understanding how the data from all these different disciplines relate to each other? How do the different levels of organization and explanation in cognitive science "talk" to each other?

I will skip over one of Bermudez's examples concerning evolutionary psychology and the psychology of human reasoning. (The quick moral is that human reasoning does *not* conform to the laws of logic or probability theory – no surprise there.)

I will also skip a difficult-to-summarize example concerning the link between (1) the ratio of oxygenated blood to de-oxygenated blood that is measured by fMRI scans and (2) single-cell neural activity.

Let us move on from such challenges of *local integration* to a larger-scale integration challenge: the search for a model that could unify *all* the ongoing discoveries of cognitive science in the way that, for example, evolutionary theory unifies all the fields and discoveries of biology.

Intertheoretic reduction

One approach to such unity is the model of *intertheoretic reduction*, which holds out the hope that high-level behavioral and cognitive processes might eventually be *reduced,* by way of *bridge laws*, to simpler events that occur at the scale of molecules in the brain.

The paradigms for such successful reduction can be found in the physical sciences. For example, the higher-level concept of *temperature*, we discovered, can be reduced to the energy of molecules. In particular, we discovered the bridge law for this, which states that the temperature of a substance is equal to the average kinetic energy of the molecules that make it up.

A triumph of intertheoretic reduction in cognitive science would look something like this: Perhaps we find that the phenomenon of vision can be reduced to a series of cognitive processes carried out by different parts of the brain, each of which can in turn be reduced to a series of sub-processes that are handled algorithmically by a set of neural networks, each of which can be reduced to neurons firing by way of synapses and neurotransmitters, each of which can be reduced to electro-chemical reactions that are already understood by chemists and physicists. A successful reduction of this kind would show that a high-level process like vision can be fully explained in terms of low-level processes described by chemistry and physics.

One step in reduction is *functional decomposition*, which decomposes a high-level process into a set of lower-level processes, each which performs a certain function. For example, psychologists have discovered that memory is not a single, unified process but rather involves three distinct sub-processes: *registering* information, *storing* that information, and then *retrieving* the information later.

We can decompose the process of memory even further by recognizing that it can involve both short-term memory (STM) and long-term memory (LTM). We discovered these were separate things because each can exist without the other. Some brain-damaged patients cannot repeat strings of numbers or words shortly after being presented with them, but they can still recognize faces and remember novels read long ago. Other brain-damaged patients can repeat strings of numbers or words without difficulty, but cannot recall information from a few minutes or more before.

Decomposition continued as we discovered that LTM, for example, involves at least two kinds of information storage: time-dependent *episodic* memory, and more abstract *conceptual* information. The brain appears to store these two types of information differently.

The goal of intertheoretic reduction would be to understand the process of memory into smaller and smaller parts until the parts under study are so small that we can actually map the neural network that performs their functions.

Mental architectures

Another model of unifying cognitive science is based on Marr’s tri-level analysis of the problem, but for now let us move on to another approach: that of mental architectures.

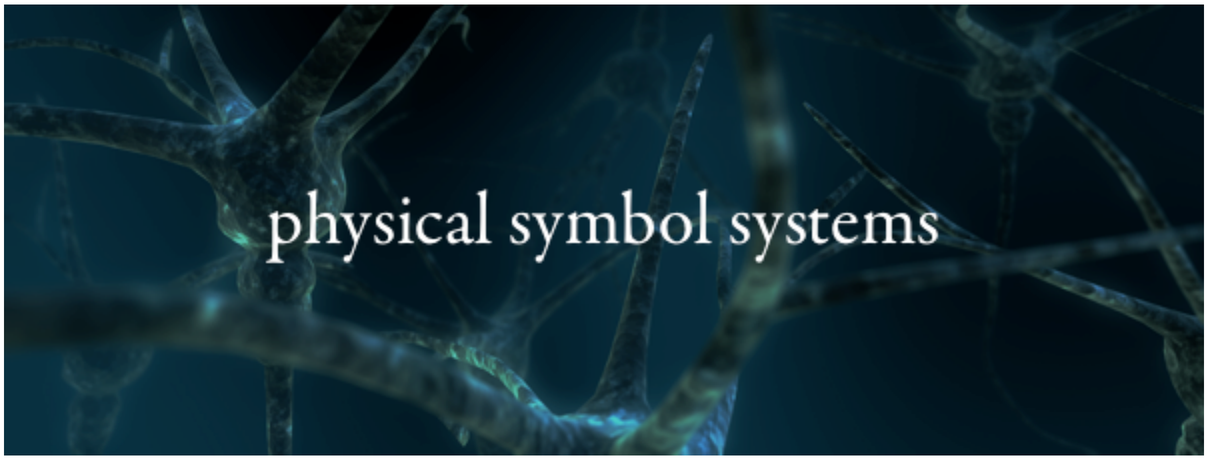
To understand this approach, consider three of the central questions of cognitive science:

1. In what format does a particular cognitive system carry information?
2. How does a particular cognitive system transform information?
3. How is the mind organized so that it can function as an information processor?

A mental architecture involves (1) a model of how the mind is organized into cognitive sub-systems and (2) an account of how information is processed in and between these different sub-systems. For such an approach to succeed, one need not discover bridge laws between the high-level phenomena like memory and the low-level systems of chemistry. Instead, one need only discover how cognitive systems process information and communicate with each other.

The mental architectures approach describes how most cognitive science is done today.

The next section covers some models which propose to answer the first two questions above: In what format does a particular cognitive system carry information, and how does it transform information? The section after that covers the third question: How is the mind organized with different cognitive systems that work together?



Though no longer "the only game in town," the idea that the mind is a computer is still popular.

In 1975, Herbert Simon and Allen Newell proposed:

> *The physical symbol system hypothesis*: A physical symbol system has the necessary and sufficient means for general intelligent action.

Their first claim was that nothing is capable of intelligent action unless it uses a physical symbol system. (Thus, the human mind must be a physical symbol system.) The second claim was that there is no in-principle obstacle to building an artificial mind, as long as you build it from a physical symbol system.

What is a physical symbol system? Well, a *symbol* is a physical pattern, and these symbols can be combined to form complex symbol structures. A physical symbol system contains these complex symbol structures, along with processes for manipulating them, and these processes can themselves be represented by symbol structures.

This is how computer programs work, and Simon & Newell proposed that the human mind works in the same way.

One fundamental concept in using intelligence (as a physical symbol system) is that of a *search-space*. Consider a half-completed chess game. The next player has a search-space of possible moves, from which he or she attempts to pick the move most likely to lead to winning the game.

More generally, a search-space is made up of all the states that can be reached by symbol transformations that are allowed. (Perhaps a state X is possible, but it would require that White's king jump to the other side of the board, and this is not an allowed "symbol transformation" in chess.)

Solving a problem means specifying a solution state (the opponent's king is checkmated) and then achieving that solution state. This is done by searching through the search-space until a solution state is found.

But how this search is done can vary. At present, our fastest computers cannot search through all possible moves and counter-moves of a new chess game in order to figure out what the winning move is at every step. (This *has* been done for checkers and many other games, though.)

Because we can't yet complete such a "brute force" search through the entire search-space of a new chess game, we might instead use a *heuristic search* that tries to trim down the search-space so that not every branch of the search-space tree needs to be explored.

One example of a heuristic search is *means-end analysis*, which works in three steps:

1. Evaluate the difference between the current state and the goal state.
2. Find a transformation that reduces this difference.
3. Check if the transformation in (2) can be applied to the current state. If it can, then apply it and go back to step (1). If it can't, find another transformation in (2).

The language of thought

But the physical symbol system hypothesis is a very general theory of how intelligence works that does not render immediate predictions. How might the physical symbol system actually be implemented by the human mind? One *specific* theory about how this is done is philosopher Jerry Fodor's *language of thought* hypothesis.

According to Fodor, the symbol structures of the mind are sentences in an internal language of thought (sometimes called Mentalese). To get a feel for Fodor's hypothesis, imagine that you are walking along a road when suddenly you hear a cry of "Help!" from behind you. You turn around because you want to help someone in need.

Now, what just happened? The auditory information received by your brain did not *force* you to turn around. Some people would have received the same auditory information and *not* turned around. So your reaction depends on how your brain processed that information.

The symbols of information carried from your ear into your brain are representations of a certain sound, and the mind processes them into symbols that represent something like a *belief* that someone behind you has called out for help. This belief interacts with another representation we might call your *desire* to help people in need if it's not too inconvenient, along with, for example, your belief that turning around is the first step toward satisfying that desire. This might generate an *intention* to turn around and go help the person who cried "Help!" And that might lead to the motor instructions that generate the movement of turning around to help.

Though we may not normally speak about beliefs and desires and intentions in terms of mental representations in a physical symbol system in the brain, we are certainly accustomed to explain people's behavior with reference to beliefs, desires, and intentions. You predict that if you race by the police car at 90mph, he will begin pursuit. Why? Because if you race by the police car at 90mph, the officer in the car will *believe* that you are speeding, he will *desire* to pursue and ticket such egregious violations of the speed limit, and thus he will form an intention to pursue and ticket you for speeding.

We are remarkably successful at predicting others' behavior by thinking about beliefs, desires, and intentions. Sometimes we make the wrong prediction, but often this is because we had the wrong information about what their beliefs and desires really were. The best explanation of the remarkable success of this *folk psychology* about beliefs, desires, and intentions, Fodor thinks, is that it is true! This folk theory about beliefs, desires, and intentions works so well because it is true. There really are mental representations in the mind that correspond to what we call beliefs, desires, and intentions. This view is called *intentional realism*.

Fodor talks of beliefs and desires *causing* action. But when beliefs and desires cause me to lift my leg, this is a different type of causation than when a doctor hits my knee with a hammer to make it bounce up. In the first case, I lift my leg because of my *desire* to begin turning around to help the person who cried out for help. My action was caused by the *content* of my beliefs and desires. The lifting of my leg was caused by, among other things, my belief *that* "Someone behind me has cried out for help" and my desire *that* "I help people who cry out for help, as long as it's not too inconvenient." It is the *propositional content* of my beliefs and desires that cause me to lift my leg.

But how can the semantic content of beliefs and desires cause action? This is the problem of *causation by content*. One reason Fodor argues for the language of thought hypothesis is that he thinks it can solve the causation by content problem.

To see why, let's begin by calling the physical properties that brains manipulate *formal properties* (properties having to do with physical *form*). In contrast, let's call the properties in virtue of which representations represent *semantic properties* (just as the branch of linguistics focused on how *words* represent is called "semantics"). Our question is: How can the brain process information if it is blind to the semantic properties of representations, and can only manipulate *formal* properties?

Consider the computer. As a Universal Turing Machine, a modern computer manipulates strings of symbols. If programmed in binary, the computer manipulates a long string of 1s and 0s (which actually are just transistors that either *do* or *don't* have electricity running through them). But the computer is oblivious to what these 1s and 0s *represent*. They might represent a sentence in a Microsoft Word document, or a block of pixels in a first-person shooter game, or wave pattern of sounds in a music file.

Even though the computer is blind to the semantic content of the 1s and 0s, it is programmed to manipulate those 1s and 0s in such a way as to yield the right result with regard to the *semantic* content of those symbols. That is why the first-person shooter game works or why Beethoven sounds so good, even though the computer has no idea what it is producing by manipulating all those 1s and 0s according to certain rules. A computer operates only on the formal properties of its physical symbols, but it does so in a way that respects their semantic properties.

And this is exactly what brains do, says Fodor. Brains and computers have to solve the same kind of information-processing problems, and so the best way to understand how the brain works is to guess that it is like a computer.

Fodor goes into more detail than this, but for now let us turn to the most famous objection to the idea that the brain is any kind of physical symbol system at all.

The Chinese room argument

Philosopher John Searle thinks that no machine built as a physical symbol system is capable of intelligent behavior, and he demonstrates his point with a thought experiment known as the *Chinese room argument*.

Imagine a person in a room. She receives pieces of paper from one window and passes out pieces of paper through a different window. The papers have Chinese symbols on them. The person has a large instruction manual that pairs input symbols with output symbols, and this tells her which symbols to pass through the output window, depending on which Chinese symbols she receives through the input window.

The instruction manual is not written in Chinese but English, and can be understood by the person in the room who knows English but not Chinese. She merely identifies the incoming symbols by their shape, finds them in the manual, and reads the (non-Chinese) instructions on which symbols to pass through the output window in response.

Now, imagine that as it happens, the inputs are all questions in Chinese, and the outputs are all appropriate answers to those questions, in Chinese. According to Searle, this thought experiment shows that it's posible for there to be successful symbol manipulation without any form of intelligence or understanding. The person in the room is giving appropriate answers in Chinese even though she has no intelligent understanding of Chinese. To someone outside the Chinese room, it will look as if the Chinese room understands Chinese intelligently. But inside the Chinese room, there is no intelligent understanding of Chinese – not in the way that a competent Chinese speaker understands Chinese.

So, Searle says, this is a counter-example to the physical symbol system hypothesis.

Responses to the Chinese room argumet

Some have said that the argument commits an equivocation. The physical symbol system is a hypothesis about the entire cognitive system. But a crucial step in the Chinese room argument is not about the system as a whole, but about one part of that system: the person in the room. Searle's claim that the Chinese room doesn't understand Chinese rests on the fact that the *person* in the room doesn't understand Chinese. Supporters of this *systems reply* argue that it is the Chinese room as a *whole* that understands Chinese and displays intelligent behavior, even though one small part of the room – the person in it – does not understand Chinese.

Searle's reply is this. Instead of imagining yourself inside the Chinese room, imagine the Chinese room being inside you! That is, imagine you have memorized the instruction manual. But that, Searle says, would not be enough to transform you from someone who doesn't understand Chinese into someone who does. What you've memorized isn't Chinese, but a long list of rules for transforming some symbols you don't understand into other symbols you don't understand.

Another reply to the Chinese room argument is the *robot reply*. A supporter of this reply would agree with Searle that the Chinese room doesn't understand Chinese, but he would say this has nothing to do with an impassable gap between syntax (symbol manipulation) and semantics (meaning). The problem is that it is not disembodied cognitive systems that understand Chinese, but embodied agents who understand Chinese. Understanding Chinese involves the ability to carry out Chinese instructions, to coordinate with other Chinese speakers, and so on. To build a machine that could do all that, you'd need to embed the Chinese room in a robot, providing it with sensory organs, a vocal system, limbs, and so on. It is this *embedded system* that would truly understand Chinese, not a disembodied Chinese room.

But again, Searle is not impressed. If this robot encounters a Chinese "stop" character on a sign, it might stop, but that's because of something it has learned to do. It no more understands the meaning of the character than does a pigeon trained not to peck at a playing card with a particular character on it.

ID3: an algorithm for machine learning

One final objection to the Chinese room argument is that despite its intuitive force, scientists have gone ahead anyway and used the physical symbol system hypothesis to great effect in building intelligent or semi-intelligent machines.

We encountered one such (virtual) machine earlier. SHRDLU was able to manipulate incoming symbols to translate them into meaningful instructions that it could then respond to or carry out in its micro-world.

Much of the field of artificial intelligence (AI) consists of *expert systems* research, in which researchers try to write programs that can reproduce or surpass the performance of expert humans on a specific task. One example of an expert system is a chess program, some of which can now reliably beat the best human chess players in the world.

One expert system is MYCIN, developed in the early 1970s to diagnose infectious diseases by taking input from doctors about a patient's symptoms, medical history, blood tests, and so on. One study showed that MYCIN produced a good diagnosis 69% of the time, which is better than the human experts who were using the same information to diagnose the patients.

Expert systems are common in the financial services industry. A simple example is an online "wizard" that guides mortgage applications through a decision tree ending in the applicant's "mortgage-worthiness." One answer to the first question leads to one branch of the tree, and thus determines what the next question will be, and so on. The rules for the decision tree might look something like this:

the next question will be, and so on. The rules for the decision tree might look something like this:

> IF income less than $40K THEN no loan.
>
> IF income greater than $75K AND no criminal record THEN loan.
>
> IF income between $40K and $75K AND applicant working for 1-5 years AND credit not good THEN no loan.

And so on. But how does this list of rules, the decision tree, get designed in the first place? One way to do it would be to gather together the mortgage loan officers at a bank and have them write out a decision tree that approximates the practices at their bank. A programmer could then reproduce this decision tree in a computer program. But this is rather boring. The real expert system is the group of mortgage loan officers, who have been reproduced in a computer program. What would be more interesting from an AI point of view would be a program capable of coming up with its own structure for solving a problem, and perhaps solving it even better than humans can.

Another approach is to get a computer program to *learn* how to build its own decision tree from a huge database of previous loan applications, their personal information, and the decision that was eventually made by the loan officers. The program would look for patterns in the data that correspond to loans being granted or denied. This would be a type of *machine learning*.

Ross Quinlan developed an influential machine learning algorithm called ID3. For ID3 to work, the database has to be set up within certain constraints. More advanced machine learning algorithms have fewer constraints, but they still have constraints of their own.

ID3 works on a database of *examples* (e.g., loan applicants). Each example has a set of *attributes*, and each attribute has a *value* associated with it. In our mortgage loan example, one attribute might be *CreditHistory*, and the assigned value for a particular example (loan applicants) could be either *Good* or *Bad*. Another attribute would be *Income* and the value assigned would be the loan applicants annual income. One attribute in particular is named as the *target attribute*; in our example this would be *Loan*, with the possible values of *Granted* or *Denied*.

ID3 runs through the database, associating the target attribute with the other attributes, and building a decision tree in which the first node might be *Income* and the final node is *Loan*. In this sense, ID3 *learns* from a huge database of past loan applicants how to build a decision tree that will roughly reproduce the actual decisions made by loan officers.

How, exactly, does ID3 turn a database into a decision tree?

> The ID3 algorithm exploits the basic fact that each attribute divides the set of examples into two or more classes. What it does is assign attributes to nodes. It identifies, for each node in the decision tree, which attribute would be most informative at that point. That is, it identifies at each node with attribute would divide the remaining examples up in the most informative way.

How do we define "most informative"? The measure used is *information gain*, which is in turn defined in terms of *entropy*. (However, "entropy" in information theory has a different meaning than it does in physics.) Entropy is a measure of uncertainty.

> Imagine that you are about to pick a ball from an urn. You know that all the balls in the urn are black or white, but you can't see the color of the ball before you pick it. Let's say the attribute you are interested in is *Black?* – i.e. whether the ball is black or not. How uncertain are you? And, relatedly, how much information would you acquire by picking a ball and seeing that it is black?

If you knew for sure that all the balls in the urn were black, your uncertainty about the color of the ball you would pick from the urn would be 0. Your entropy (in the sense of information theory) would be 0.

But in this case, half the balls are black, and half the balls are white. One is just as likely as the other. So you are in a state of *maximum* uncertainty about the color of the next ball you'll pick, which is defined as an entropy of 1.

What if you knew 60% of the balls are black, and 40% are white? Then your uncertainty is less than 1. That is, you can be *slightly* confident that the ball you will pick is black, though not as confident as you would be if you knew that 98% of the balls in the urn were black.

> Once we have a formula for calculating entropy we can calculate information gain relative to a particular attribute. Since we are trying to measure information *gain*, we need to work out some sort of baseline. The ultimate aim of the algorithm is to produce a decision tree in which each branch ends in a value for the target attribute (i.e. the loan application is either accepted or declined). It makes sense to start, therefore, by considering the entropy of the total set of examples relative to the target attribute. We can call this set S. Calculating the entropy of S measures our degree of uncertainty about whether examples in S will have the target attribute.

Roughly, the ID3 algorithm checks each attribute for how much entropy would be reduced if the whole set of examples was classified according to that attribute. This would be the attribute with the highest information gain. ID3 assigns the attribute with the highest information gain as the first node in the decision tree.

After the first node we have at least two branches. ID3 repeats the procedure on the subset of examples contained in one branch of the tree, and assigns the attribute with the highest information gain along that branch as the next node in that branch. It then does the same for all remaining branches and sub-branches until the decision tree is built, ending with the target attribute (i.e. whether the loan is approved or denied).

One important use of ID3 came from Ryszard Michalski and Richard Chilausky, who used ID3 to diagnose diseases in soybeans. This was a more complicated problem than a mortgage loan application. There are 19 common diseases among soybeans, so the target attribute *Disease* had not *two* possible values but *nineteen*. Moreover, experts used at least 35 different symptoms of disease. After writing the program and letting it learn from a database of examples, they compared the effectiveness of their program to the judgments of actual experts in plant disease. In a sample of 376 cases, the human experts had an 87% success rate, but the ID3 machine learning program had a 99.5% success rate success rate.

Machine learning programs are one example of physical symbol systems that are often capable of out-performing intelligent humans on certain tasks. But the physical symbol system hypothesis need not be realized with such language-like symbol structures. They can also work on *imagistic* data, as the brain apparently does in some cases (recall the discussion of mental rotation above).

WHISPER

A physical symbol system might be implemented with language-like symbols like those in Fodor's language of thought hypothesis. But it might also be implemented with more image-like *diagrammatic representations*. A computer program named WHISPER, designed by Brian Funt, was written to work with diagrammatic representations.

WHISPER runs in a virtual world containing blocks of many shapes and sizes, all sitting on a flat surface. The job of WHISPER is to assess the stability of the structures of blocks and predict how they will collapse.

WHISPER contains a high-level reasoner (HLR) at the "top" of the system. It is programmed with an understanding of the simple physics of the block world. It gets information about the block world through a virtual retina, through which it "sees" the initial problem state: a particular pile of blocks. It then produces a series of "snapshots" showing how the pile of blocks will collapse according to the laws of physics in this block world, and stops when it produces a diagram in which all the blocks are stable – the collapse has ended.

Here is WHISPER's basic algorithm, assuming that objects move only by *rotating* (allowing them to *slide* as well introduces another layer of complexity):

1. Determine all instabilities.
2. Pick the dominant instability.
3. Find the pivot point for the rotation of the unstable object.
4. Find the termination condition of the rotation using retinal visualization.
5. Call transformation procedure to modify the diagram from Step 4.
6. Output modified diagram as a solution snapshot.
7. Restart from Step 1 using diagram from Step 6 as input.

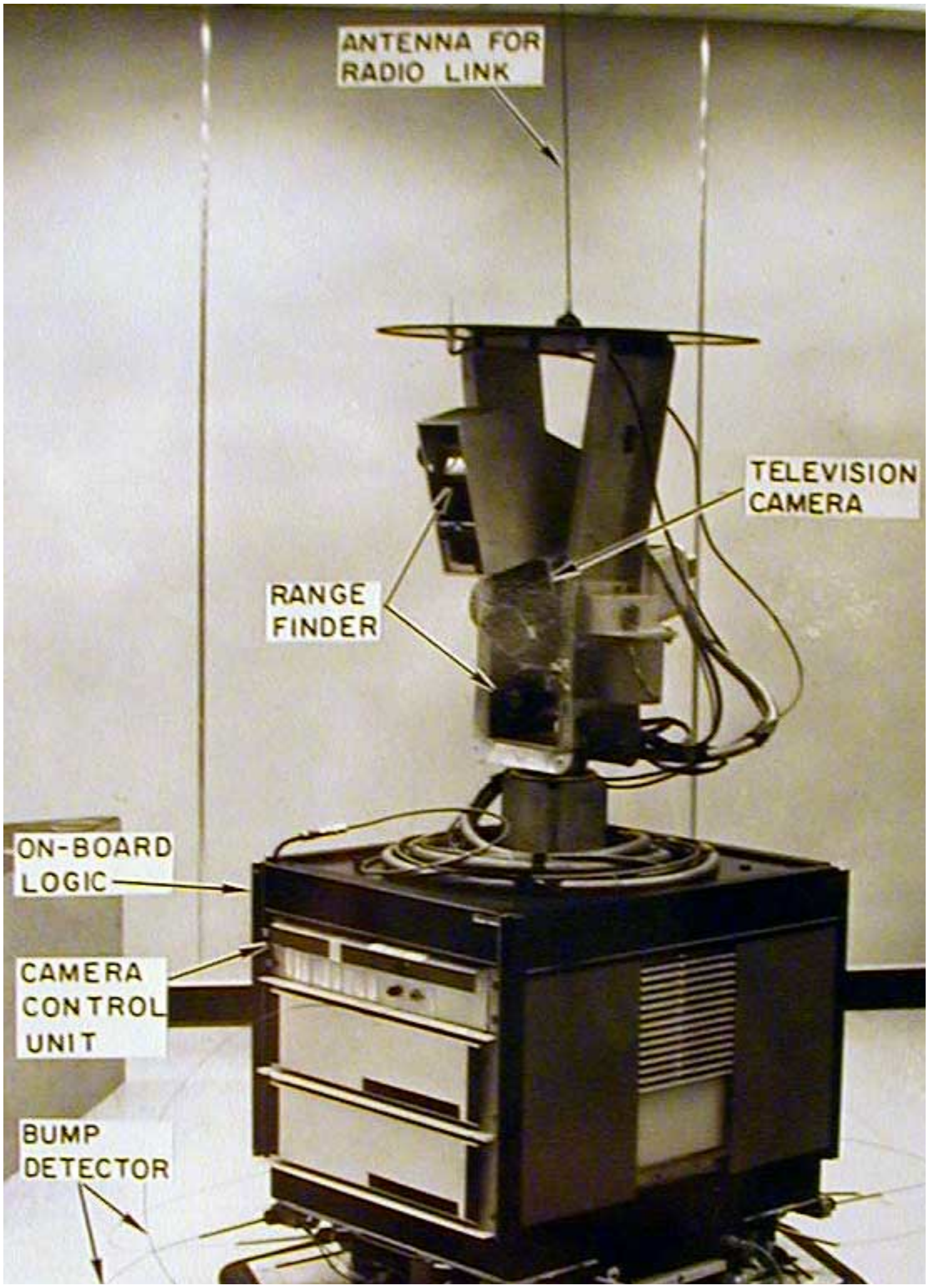
By this method (which is far more detailed in the actual program, of course), WHISPER solves a particular *chain reaction problem*. With such calculations, it can "see", snapshot by snapshot, what will happen as the blocks collapse upon one another according to the simplified laws of physics in its block world.

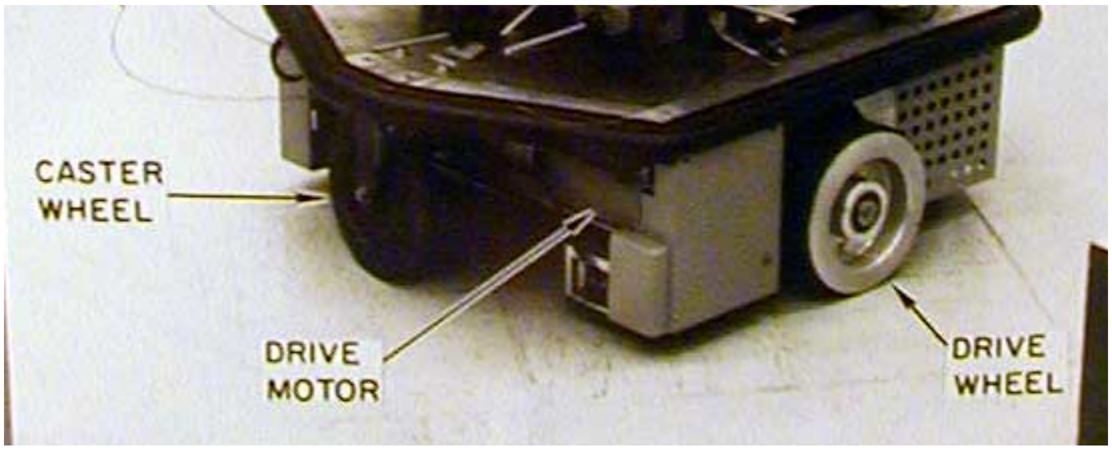
WHISPER illustrates how broad the physical symbol system hypothesis can be, for it processes symbols diagrammatically (in an image-like way) rather than propositionally (in a language-like way).

SHAKEY the robot

So far we have discussed intelligent "action" only in a loose sense. ID3′s "action" is to organize a database into a decision tree. WHISPER's "action" is to predict how a structure of virtual blocks will database into a decision tree. WHISPER's "action" is to predict how a structure of virtual blocks will collapse. Now, we look at a physical symbol system capable of a richer type of intelligent action: intelligent action in a real, physical environment.

SHAKEY, developed in the early 1970s, was the first robot able to perceive its environment and implement complex instructions in a physical environment:





SHAKEY was placed in a building with a series of rooms. Each room had a door or two leading to other rooms, each door clearly labeled with the names of the rooms it connected. There were also blocks set down as obstacles in each room. The huge computer running SHAKEY's software was actually located elsewhere, and communicated with SHAKEY via a radio link.

SHAKEY's software is a heirarchical structure of complex behaviors that are broken down into simpler and simpler behaviors and calculations, with a specific program to run each simple behavior and calculation. One level of the heirarchy has programs for implementing Low-Level Activities (LLAs). LLAs include rolling forward and backward, taking a photo with its onboard camera, tilting its "head" up or down or panning it from side to side, and so on.

SHAKEY's software also contains a model of its environment (except for the contents of the Mystery Room, which are unknown until SHAKEY interacts with it), and a model of where SHAKEY is in that environment and what state SHAKEY is in. Some of this information comes from its visual system and its whisker-like sensors near its rollers on the floor.

The hierarchical organization of SHAKEY's software allows SHAKEY to perform complex tasks. For example, the programmers can tell SHAKEY to push a particular block from one room into another. One set of programs translates this command into a set of actions that SHAKEY will need to take, using input about where SHAKEY is, where the target block is, and where the target room is. Another set of programs execute these specific actions.

For example, one task is to figure out which room SHAKEY is currently in. SHAKEY knows which blocks are in each room, so its first task is to move around the room it's in, take photographs and bump into things, and from this deduce what blocks are in the current room, and thus which room it is in. After that, SHAKEY can consult the map of the building in its head to figure out which path would lead most quickly to the room where the target block is, by way of a heuristic search (discussed above). Various programs in SHAKEY's software will then break this task of moving to the room with the target block into one small movement after another, moving around obstacle blocks encountered in the environment, and so on.

SHAKEY is an early example of how a physical symbol system is capable of intelligent behavior in the real world.

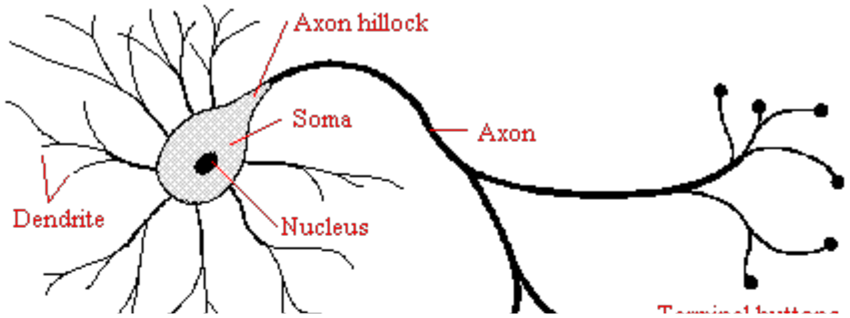
The physical symbol system hypothesis is modeled after computers, but a new model of information processing called a *neural network* is inspired by an idealized model of how neurons work.

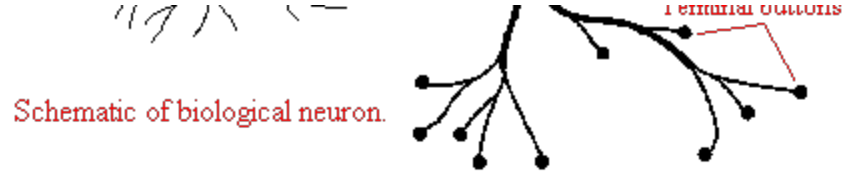
We saw earlier that we have well-developed techniques for measure brain activity at a large scale (with fMRI, for example), and to measure brain activity at the level of individual neurons (through single-cell recordings), but our understanding of how the brain carries out its functions at the *systems* level (networks of many neurons) is still weak.

We don't yet have the equipment to study populations of neurons directly, in detail. So, researchers try to study populations of neurons *indirectly*, by developing models that approximate important features of populations of neurons. These models are neural networks.

As discussed earlier, neural networks are made of individual units modeled after neurons. Each neuron has a cell body (a *soma*) containing a *nucleus*, and there are many root-like extensions leading away from the cell body. Most of these extensions are *dendrites*, and one of them is a thicker *axon* which eventually splits into several branches, each terminating in an *endbulb* (or *terminal button*) that comes up next to the dendrites of another neuron. The gap between an endbulb of one neuron and a dendrite of a different neuron is called a *synapse*.

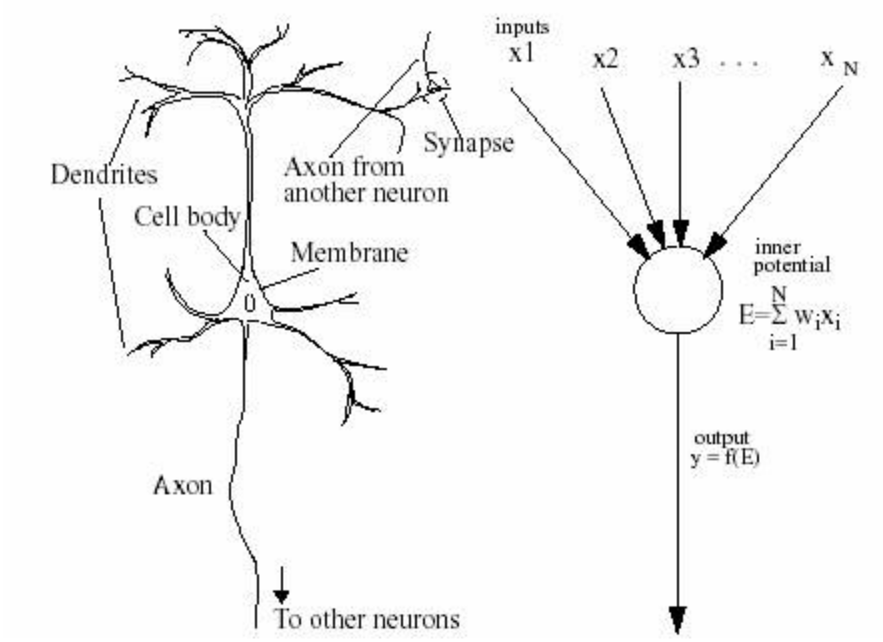
Dendrites are the *receivers* of a neuron. Dendrites receive signals from other neurons, between 5,000 and 50,000 of them. A neuron transmits the signal along its axon to a synapse at the other end. When the signal from the sending neuron (or *presynaptic* neuron) reaches the synapse, it generates an electrical signal in the dendrites of the receiving neuron (or *postsynaptic* neuron), which begins the process again in the next neuron (along with the other signals being received by the dendrites of that neuron).





Whether or not a neuron actually fires an electrical signal along its axon depends on the input received by its dendrites. Some of the signals received by the neuron's dendrites promote this firing, and others inhibit it. That is, some synapses are *excitatory*, and others are *inhibitory*. Imagine that each excitatory synapse has a positive weight, and each inhibitory synapse has a negative weight. We can calculate the strength of each synapse by multiplying the strength of the incoming signal with the negative or positive weight of that synapse. You then add up all these synaptic strengths (some positive, some negative) and get the total input strength of the signal being received by all the neuron's dendrites. If this total input strength exceeds the neuron's *threshold*, then the neuron will fire a signal along its axon.

Neural networks are built with units or *artificial neurons* meant to imitate how biological neurons work. An artificial neuron receives inputs of various strengths. If these negative and positive inputs add up to be greater than the threshold of the artificial neuron, then a total output for that neuron is calculated and a signal of that strength is sent as the output signal.



The output signal need not be identical to the received input. The *activation function* of a biological or artificial neuron might be a *threshold linear* function like this:

> IF input level is less than 60, send no output signal. IF input level is greater than 60, THEN release an output signal that is 3.5 times greater in strength than the input level.

Or, it might have a binary threshold function like this:

> IF input level is less than 60, send no output signal. IF input level is greater than 60,

> IF input level is less than 60, send no output signal. IF input level is greater than 60, THEN release an output signal of level 80.

Or, it might have a *sigmoid* function, according to which the strength of the output signal varies with the strength of the input signal, but also has a threshold below which total input has little effect, and a ceiling above which additional input also has little effect.

This is how an individual unit in a neural network functions. How then does an entire neural network work? Let us consider a simple type of neural network, the *single-layer network*.

Single-layer networks

Think of information processing in terms of *mapping functions*. A function simply converts a given set of inputs into a definite output. Addition is a function. Given two *inputs*, the addition function gives us a third number as its *output*. For any given set of inputs, a function has one possible output.

All the sets of possible inputs to a function are called its *domain*. The addition function has a very large domain. In its domain are the following sets: {1, 5}, {18, 19.2, 1/3}, {21, 0, 4, 18,000,004, -98}, and many others. All the possible outputs of a function are called its *range*. A function maps each particular set of possible inputs from its domain to a particular, single output in its range. For example, the addition function maps the input set {1, 5} to the output of 6.

One class of functions is called *binary Boolean functions*. A binary Boolean function has only two possible outputs in its range (what we might think of as TRUE and FALSE), and instead of having sets of numbers in its domain, it has only pairs of truth values.

Consider how we might formalize one use of the English word "and" as a binary Boolean function. Let's say we have a statement A, which can be true or false, and a statement B, which can also be true or false. The binary Boolean function for "and" tells us whether "A and B" is true or false, depending on whether A is true or false and on whether B is true or false.

The "and" function with inputs A and B always gives a definite output for "A and B." If A is false and B is false, the function would tell us that "A and B" is false. If A is false and B is true, the function would tell us that "A and B" is still false. If A is true and B is false, the function would tell us again that "A and B" is false. Only if A is true and B is true does the function tell us that "A and B" is true.

To represent this AND function as a single-layer network, we'll represent TRUE with the number 1, and FALSE with the number 0. Now, remember our goal. We want our AND function to output 0 whenever the input is either {0, 0} or {0, 1} or {1, 0}, but the output should be 1 if the input is {1, 1}. With this setup, our activation function can be rather simple:

> IF *Input1* + *Input2* = 2, THEN fire output signal of strength 2.

In this way, a single-layer network can represent the Boolean AND. By similar means, it can also represent the Boolean NOT, and the Boolean OR.

By chaining together simple AND-gates, NOT-gates, and OR-gates, computer scientists can do anything a computer can do. A computer can be used to simulate a neural network, but a neural network could also be programmed to simulate a computer.

But how do the weights get set, and how does the threshold get set? Could the weights and thresholds be *learned*?

Learning in single-layer networks

In 1949, Donald Hebb speculated about how the brain might learn:

> When an axon of a cell A is near enough to excite cell B or repeatedly or persistently takes part in firing it, some growth or metabolic change takes place in both cells such that A's effeciency, as one of the cells firing B, is increased.

This learning process, called *Hebbian learning*, occurs by synaptic modification. Every time B fires after A, this increases the probability that B will fire after A fires, which is what Hebb meant by an increase in A's "efficiency." Hebbian learning occurs by strengthening the *association* between two neurons. The slogan is:

> Neurons that fire together, wire together.

This is a kind of "unsupervised" learning (rewiring), and there is much evidence that it occurs in the brain.

In the 1950s, Frank Rosenblatt developed a method of "supervised" learning in *artificial* neural networks. Rosenblatt wanted to find a learning rule that would allow a network with randomly-assigned weights and a randomly-assigned threshold to settle itself on a configuration of weights and threshold that would allow it to solve a given problem – that is, produce the correct output for every input.

Bermudez explains:

> …whenever the network produces the wrong output for a given input, this means that there is something wrong with the weights and/or the threshold. The process of learning (for a neural network) is the process of changing the weights in response to error. Learning is successful when these changes in the weights and/or the threshold converge upon a configuration that always produces the desired output for a given input.

This kind of learning is "supervised" in that it is responsive to whether or not the "correct" output was achieved.

We can understand how it works *mathematically* with a bit of symbolization. Consider the AND-gate we discussed earlier. When both inputs have a value of 1, the output should be 1. If any of the inputs is a 0, the output should be 0.

But let us imagine the weights and threshold of the neural network are first assigned randomly. Then the actual output is probably not going to be the intended output every time.

Let's label the discrepancy between the actual and intended output with a δ (small delta). So:

δ = IntendedOutput – ActualOutput

We also need to specify a *learning rate*. How much are we going to change the weights and threshold on each trial? We label the learning rate with ε (small epsilon), which has a value between 0 and 1.

Finally, we'll use the symbol Δ (big delta) to indicate the adjustment to be made after each trial. T is the treshold, I_i is the *i*-th input, and W_i is the weight attached to the *i*-th input. Given this, Rosenblatt's learning algorithm for a developing an AND-gate might be:

$\Delta T = -\varepsilon \times \delta$

$\Delta W_i = \varepsilon \times \delta \times I_i$

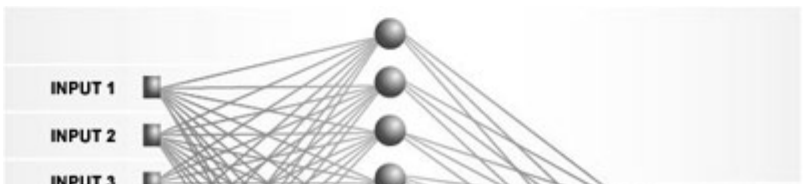
Bermudez explains:

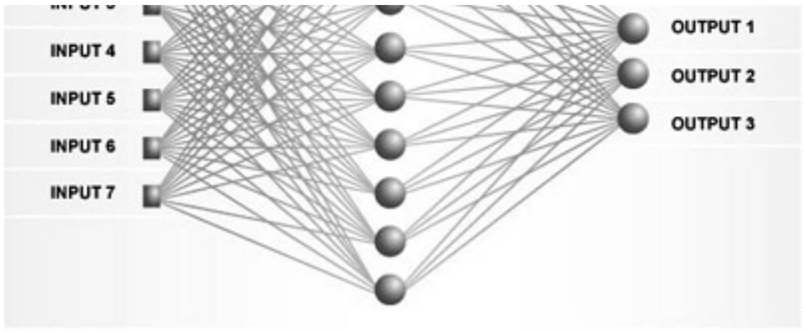
> One obvious feature is that the two changes have opposite signs. Suppose δ is positive. This means that our network has undershot (because it means that the correct output is greater than the actual output). Since the actual output is weaker than required we can make two sorts of changes in order to close the gap between the required output and the actual output. We can *decrease* the threshold and we can *increase* our weights. This is exactly what the perceptron convergence rule [Rosenblatt's learning technique] tells us to do. We end up decreasing the threshold because when δ is positive, $-\varepsilon \times \delta$ is negative. And we end up increasing the weights, because $\varepsilon \times \delta \times I_i$ comes out positive when δ is positive.

Rosenblatt's "perceptron convergence rule" is powerful. In fact, it has been proved that applying the rule is guaranteed to converge on a solution in every case where a solution exists. But for many problems, a solution does not available to a single-layer network. What is needed for some problems is a learning algorithm that works in multilayer networks.

Multilayer networks

A multilayer network contains *hidden layers*, so called because they have no direct contact with input or output – they are "hidden" from the outside world. Input signals are received by the units in the input layer. These units are not connected to each other, but each unit in the input layer *is* connected with every unit in the next layer, and so on through all the hidden layers to the output layer. Each unit in the output layer outputs a certain level of activation to whatever it is connected to. Each connection between two units has a given weight, and each unit has a given threshold, which when reached will cause the unit to pass on the signal according to its activation function. The input to a given unit is, just as before, the sum of all the activation levels that reach it.





Recall the multilayer neural network that was able to detect mines with sonar, discussed above. It learned by a method called *backpropagation of error*. How does that work, exactly? Bermudez explains:

> The basic idea is that each hidden unit connected to an output unit bears a degree of "responsibility" for the error of that output unit. If, for example, the activation level of an output unit is too low, than this can only be because insufficient activation has spread from teh hidden units to which it is connected. This gives us a way of assigning error to each hidden unit. In essence, the error level of a hidden unit is a function of the extent to which it contributes to the error of the output unit to which it is connected. Once this degree of responsibility, and consequent error level, is assigned to a hidden unit, it then becomes possible to modify the weights between the unit and the output unit to decrease the error.
>
> This method can be applied to as many levels of hidden units as there are in the network. We begin with the error levels of the output units and then assign error levels to the first layer of hidden units. This allows the network both to modify the weights between the first layer of hidden units and the output weights and to assign error levels to the next layer of hidden units. And so the error is *propagated* back down through the network until the input layer is reached.

Biology and neural networks

But how biologically plausible are neural networks? There are some obvious differences between neural networks and the brain. Neural network units are homogenous, but there are many different types of neurons in the brain. The brain is not nearly as massively parallel as a typical neural network. Moreover, there is no evidence that anything like backpropagation of error occurs in the brain. Finally, most learning in neural networks occurs as a result of specific feedback about the difference between an intended output and the actual output. But the brain doesn't seem to use such detailed feedback.

Luckily, computer scientists and computational neuroscientists have developed other learning models that may be more biologically plausible. These are known as *local algorithms*. In local algorithms, a unit's weight changes according to the inputs to and outputs from that unit, as in Hebbian learning. Another type of learning algorithm is used in *competitive networks*, which I won't discuss here.

Differences between physical symbol systems and neural networks

In a physical symbol system, the structure of a complex symbol is directly correlated with the structure of the information it is carrying. There are some neural networks, called *localist* networks, for which this is true, but most neural networks are *distributed* networks. The information in a distributed network is not located in any specific place. Rather, the information is encoded as a pattern of weights that are distributed across all the connections in the network.

In a physical symbol system, information is stored as symbols, and processing occurs by way of rules for manipulating those symbols. There is a difference between the representations on which the rules operate and the rules themselves. But in a neural network, there is no comparable distinction between information storage and information processing. Both are contained in the pattern of weights of the connections in the network.

Physical symbol systems must operate according to fixed rules, and can even "learn" according to these fixed rules, but neural networks learn by a very different method. They learn by changing the weights of the connections in the network.

Neural networks and language learning

On its face, human language seems to fit better with the physical symbol system hypothesis than with neural network models of the brain. After all, human language is a system of symbols (sounds, or letters and punctuation) that are put together in symbol structures (strings of sounds, or printed words and sentences) that are manipulated by very specific rules (the rules of grammar).

Jerry Fodor thinks that learning a language is a matter of mastering a set of rules in the language of thought (a physical symbol system hypothesis), and he gave a famous line of argument for this in *The Language of Thought*.

On the other hand, neural networks fare surprisingly well in learning language. Rather simple networks designed by Jeff Elman have been trained to predict the next letter in a sequence of letters, or the next word in a sequence of words.

More interestingly, language-learning neural networks have been shown to reproduce some phenomena characteristic of children who are learning languages (in particular, how they learn to use the past tense) – evidence that some theorists have taken to weigh in favor of the neural network model of language learning.

Bermudez also gives an example of how neural networks might account for how infants acquire a very basic theory of physics, but I will not cover that here.

The question of levels

Recall David Marr's distinction between three levels at which we might think of a cognitive system: the computational level, the algorithmic level, and the implementational level. Several physical symbol system theorists have pointed out that we can think of neural networks at either the implementational level or the algorithmic level. But if the mind is a neural network at the implementational, this does not rule out the physical symbol system hypothesis, but merely shows us how physical symbol systems can be *implemented*.

But can neural networks account for algorithmic processes, which are characterized by representations with separable and recombinable components? If so, it still seems they are merely *implementations* of physical symbol systems. And if not, then it's not really fair to think of them as algorithmic information processors. Either way, the neural network model is not a serious competitor to the physical symbol system hypothesis.

But perhaps this begs the question against neural networks, or perhaps we should not be thinking about the brain in terms of Marr's three levels. In any case, the answer will come from a closer examination of the brain.

In trying to design intelligent agents (systems that perceive their environment and act upon it, like SHAKEY), AI researchers use a variety of *agent architectures*: blueprints for how different modules are organized to compose the agent. Perhaps the human mind is organized like one of the popular agent architectures from AI research.

The simplest agent architecture is a *simple reflex agent*, in which signals from the sensory input systems directly determine the actions of the "effector" systems. These direct links are the result of *condition-action* rules of form *IF condition C holds THEN perform action A*. Cognitive scientists don't even call this a cognitive system, because it is not a system that processes information between its sensory input and its motor output.

A primitive kind of *cognitive* system is a *goal-based agent*. Goal-based agents do not merely react to environmental stimuli. Instead, they (1) perceive what the world is like now, (2) predict what the world will be like if certain actions are taken, and then (3) perform the action predicted to satisfy the most and strongest of its programmed goals.

Even more sophisticated is a *learning agent*. A learning agent has goals, and is able to predict which actions will satisfy those goals, but it can also detect (with a module called the Critic) mismatches between predicted goal satisfaction and action goal satisfaction, and learn to minimize the difference between the two. So, it can achieve its goals better after 20 rounds than at the beginning, by learning from past successes and failures.

The latter two agent architectures contain sensory systems, effector system, and other systems, for example the Critic and the learning system. Jerry Fodor and others have proposed that the human mind is similarly modular: made of many systems and sub-systems.

In *The Modularity of Mind*, Fodor notes that much of psychology assumes that mental faculties are *domain-general* in that they can call upon all types of information. For example, any data can be retained and recalled in memory, and any perception is a candidate for attention. But, Fodor suggests, perhaps cognitive systems are instead *domain-specific*. Perhaps there are systems only for very specific tasks like analyzing shapes and for recognizing faces, and the types of information available to these systems is limited (they are *informationally encapsulated*). These limited systems are what Fodor calls *cognitive modules*. These modules, says Fodor, also suffer from *mandatory application* in that they automatically respond to stimuli of the appropriate kind, without waiting for commands from a central processor.

Fodor thinks the following are likely candidates for cognitive modules: color perception, shape analysis, analysis of three-dimensional spatial relations, face recognition, grammatical analysis of heard utterances.

But, he says, there must also be a "central processor" capable of evaluating and correcting the output of various cognitive modules. The central processor might be focused on building a true model of the world within the mind it serves, for example. The central processor interacts across a wide variety of modules, and is thus non-modular itself.

But Fodor is pessimistic about our ability to understand the central processor, because "the more global… a cognitive process is, the less anybody understands it." Cognitive science, he thinks, is best suited to understanding individual modular processes.

Massive modularity

Others have thought that there is no central processor. Rather, *all* mental processing is modular. This is the *massive modularity hypothesis*.

Evolutionary psychologists, for example, have suggested that the human brain contains a large number of modules that were especially useful for our hunter-gatherer ancestors. These *Darwinian modules* might include modules for face recognition, emotion detection, gaze following, folk psychology (the beliefs and desires of others), cheater detection, and so on.

Another module might be one for folk physics. Infants, for example, appear to be pre-programmed with some basic mechanical expectations.

More evidence comes from brain impairments. Those who suffer from *prosopagnosia*, or face blindness, cannot recognize faces, even though they can recognize other types of objects. Prosopagnosia is associated with injury to a very specific brain area, now called the fusiform face area.

But why think that the *whole* brain is like this? Why think it is nothing but small modules? Two important arguments for this conclusion come from a 1994 paper by Leda Cosmides and John Tooby: "Origins of domain specificity: The evolution of functional organization."

Both arguments in that paper draw upon the assumption that any mental architecture we have must have evolved because it solved the adaptive problems of our ancestors have evolved because it solved the adaptive problems of our ancestors.

The first argument is *the argument from error*. Learning occurs by getting things wrong. Some error signals, like pain and hunger, are obvious. But what error signals could work for what goes on in central processing? Adaptive fitness criteria, they argue, are domain-specific, not domain-general, and so a domain-general cognitive system could not have been adaptive, and could not have evolved. The second argument, *the argument from statistics and learning*, is more complicated and subtle, and won't be covered here.

On the other hand, consider classical conditioning and operant conditioning. These seem to be domain-general, and yet they are a kind of learning. So perhaps we have direct evidence that domain-general learning can happen, no matter what evolutionary arguments Cosmides and Tooby can muster. Jerry Fodor and Steven Pinker have levied other arguments for why there *must* be domain-general cognitive systems in the human mind.

But perhaps mental architectures are a hybrid of a modular systems, some using physical symbol systems, others using neural networks. An example is the ACT-R/PM architecture developed chiefly by John R. Anderson, which we won't cover here. (Again, I recommend you buy the book.)

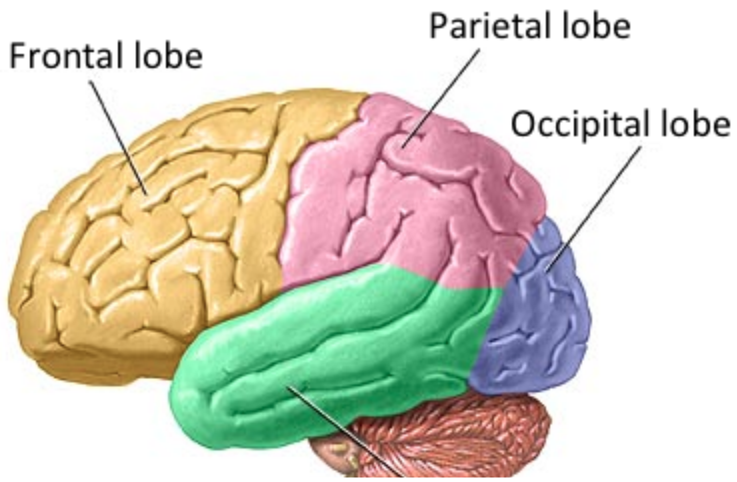
Mapping the brain

Whether they think the brain has central processes or not, most cognitive scientists think the brain is organized into cognitive sub-systems. They've tried to located these systems in the brain with various brain-mapping techniques.

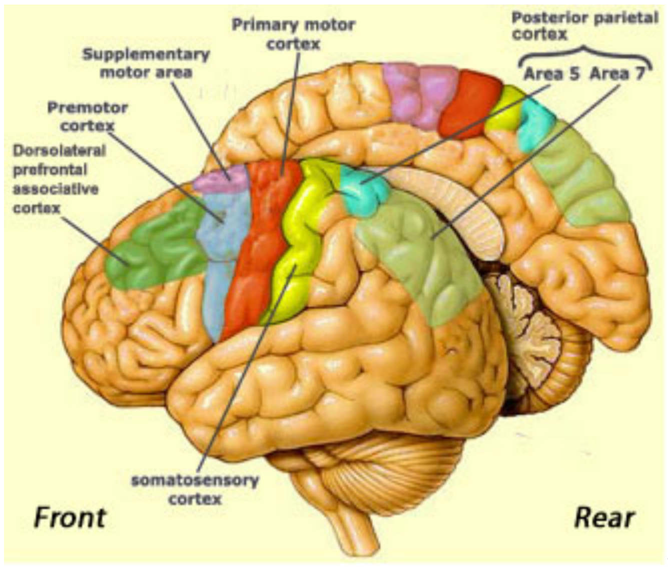
First, let us survey the regions of the brain.

The brain comes in two halves: the left hemisphere and the right hemisphere. Each hemisphere is divided into four lobes:

- the frontal lobe (for reasoning, planning, motor execution, mood control)
- the parietal lobe (integrates the sense of touch and body position with the visual system)
- the occipital lobe (for visual perception and spacial processing)
- the temporal lobe (for auditory perception, language, long-term memory and emotion)



The brain has many bumps and grooves. The bumps are called *gyri* and the grooves are called *sulci* (the singular forms are *gyrus* and *sulcus*). Sulci are also called *fissures*. Some of the larger gyri and sulci have names. I'll bet you can guess where the parieto-occipital sulcus is. Less obvious is the Sylvian sulcus, which separates the temporal lobe from the frontal and parietal lobes. It is named after a 17th-century professor of medicine, Franciscus Sylvus.

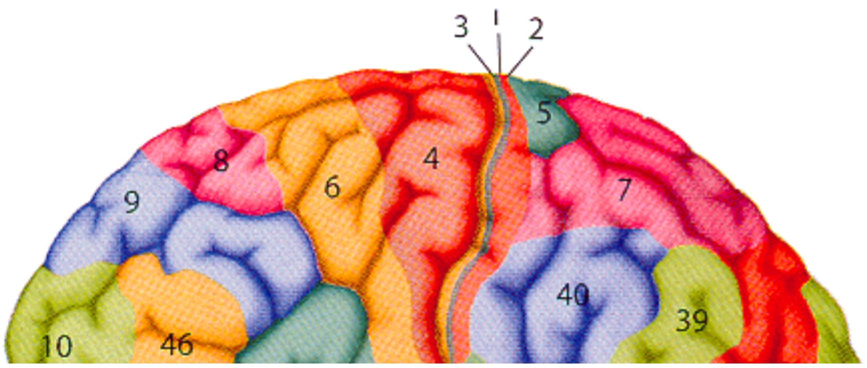


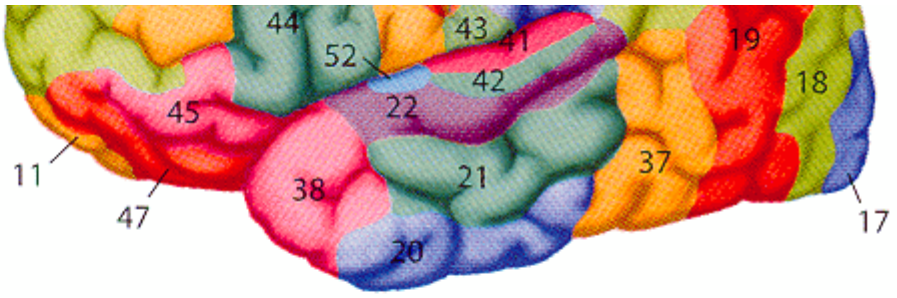
Some brain areas are named not by anatomical features, like the Sylvian sulcus, but by its supposed *functional role*, such as the *premotor cortex* or the *supplementary motor area*.

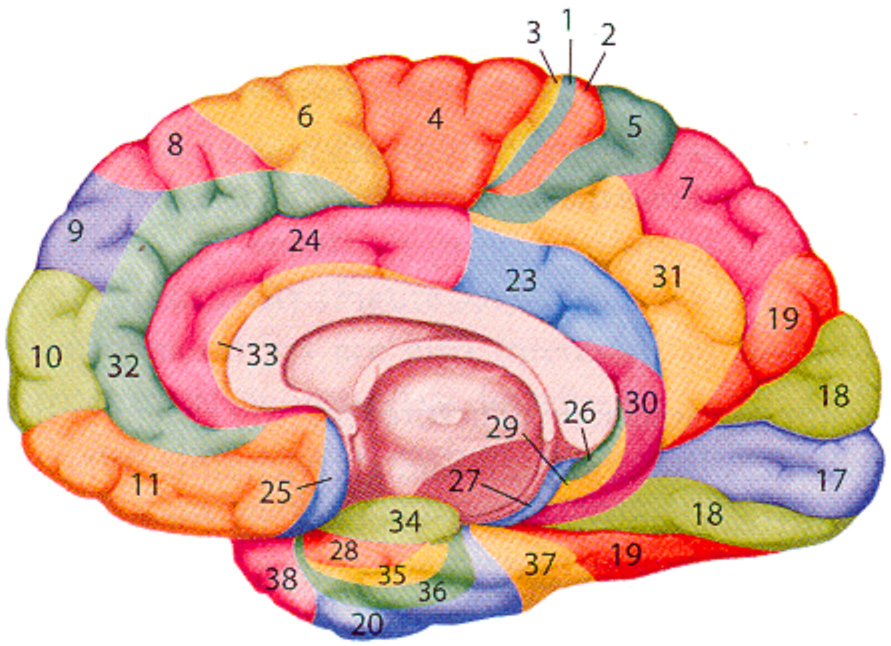
After noticing large-scale features like the lobes and major gyri and sucli, how can we identify brain areas more specifically? A common method is to use a system developed by neuroanatomist Korbinian Brodmann.

Brodmann found that brain regions can be distinguished by the types of cell they contain and how dense those types of cell are. Using this method, he numbered over 50 different brain regions, now called *Brodmann areas*.

Here they are, from two different views:







Remarkably, this classification by types of neuron and neuron density maps often maps very nicely onto *functional* regions. For example, the primary visual cortex, where retinal information is sent, is Brodmann area 17. Information gained from the touch sense and the body-position sense arrive in an area known as the somatosensory cortex, which is Brodmann area 3.

Much of neuroscience is concerned with separating the brain into areas of distinct neuronal populations.

Another major area of research concerns the study of how these segregated brain areas are connected. One technique for this is called *tract tracing*. Researchers inject a marker chemical into the body of a nerve cell, which then transmits it along its axon. Tracing where the marker ends up helps to reveal the anatomical connectivity within the brain.

Such invasive techniques, however, are only carried out on animals. Thus, we have a pretty good map of the connectivity in the brain of the macaque monkey, but not so good a map of connectivity for the human brain. Also, using tract tracing does not tell us about the *direction* of information flow, merely about the path of physical connection.

Unfortunately, we can't map cognitive activity (information processing) directly. So we must map other things.

Neurophysiologists can measure a neuron's electrical activity by placing a microelectrode next to the cell being recorded. This can be used to identify neurons sensitive to particular stimuli. For example, neuroscientists in Italy have found "mirror neurons" in monkeys that fire both when the monkey performs a specific action *and* when it observes that action being performed by someone else.

On a larger scale, electroencephalography (EEG) measures the electrical activity of thousands of neurons through electrodes attached to the skull. Magnetoencephalography (MEG) measures these same electrical currents, by way of the magnetic fields they produce. It allows for better spatial resolution than EEG does.

Another technique is to map the brain's blood flow and blood oxygen levels. PET scans map blood flow in the brain by tracking the movement of radioactive water, and fMRI measures levels of blood oxygenation, both of which are indicators of neuronal activity. These methods have fairly good spatial resolution, but very poor temporal resolution. (EEG and MEG have poor spatial resolution, but high temporal resolution.)

fMRI is most often used to identify brain areas associated with carrying out a specific cognitive task, since this requires high spatial resolution but not so much temporal resolution.

Finishing up

Bermudez includes an entire chapter on the story of how we used these methods to figure out how it is that we read other's minds (that is, how we understand the thoughts of others). Mindreading seems to be made up of at least six different systems, each emerging at different stages in development. But there is still much debate about how mindreading is accomplished.

Bermudez also includes two chapters on some recent developments in theoretical cognitive science that I will not cover. One chapter covers dynamical systems, a way of understanding how agents can respond to their environment without assuming there are internal systems that carry out specific information-processing tasks or representation.

The last chapter looks ahead to the challenges and applications of cognitive science. One exciting trend is that of neuroprosthetics – the development of artificial brain systems. Some neuroprosthetics, such as cochlear implants that can cure certain types of deafness, are already in wide use.

To explore cognitive science further, I recommend the following links:

- Companion site to Bermudez' *Cognitive Science*, with tons of useful links
- Paul Thagard's overview of cognitive science
- Other introductory books on cognitive science: Thagard's *Mind*, Friedenberg & Silverman's *Cognitive Science*, Kolak et. al.'s *Cognitive Science*, Stillings et. al.'s *Cognitive Science*

Enjoy!