1 MEAT MACHINES
Mindware as Software

1.1 Sketches

The computer scientist Marvin Minsky once described the human brain as a meat machine—no more, no less. It is, to be sure, an ugly phrase. But it is also a striking image, a compact expression of both the genuine scientific excitement and the rather gung-ho materialism that tended to characterize the early years of cognitive scientific research. Mindware—our thoughts, feelings, hopes, fears, beliefs, and intellect—is cast as nothing but the operation of the biological brain, the meat machine in our head. This notion of the brain as a meat *machine* is interesting, for it immediately invites us to focus not so much on the material (the meat) as on the machine: the way the material is organized and the kinds of operation it supports. The same machine (see Box 1.1) can, after all, often be made of iron, or steel, or tungsten, or whatever. What we confront is thus both a rejection of the idea of mind as immaterial spirit-stuff and an affirmation that mind is best studied from a kind of engineering perspective that reveals the nature of the machine that all that wet, white, gray, and sticky stuff happens to build.

Box 1.1

THE "SAME MACHINE"

In what sense can "the same machine" be made out of iron, or steel, or whatever? Not, obviously, in the strict sense of numerical identity. A set of steel darts and a set of tungsten ones cannot be the *very same* (numerically identical) set of darts. The relevant sense of sameness is, rather, some sense of *functional* sameness. You can make a perfectly *good* set of darts out of either material (though not, I suppose, out of Jell-O), just as you can make a *perfectly* good corkscrew using a myriad of (in this latter case quite radically) different designs and materials. In fact, what *makes* something a corkscrew is simply that it is designed as, and is capable of acting as, a cork-removing device. The notion of a brain as a meat machine is meant to embody a similar idea: that what matters about the brain is not the stuff it is made of but the way that stuff is organized so as to support thoughts and actions. The idea is that this capability depends on quite abstract properties of the physical device that could very well be duplicated in a device made, say, out of wires and silicon. Sensible versions of this idea need not claim then that *any* material will do: perhaps, for example, a certain stability over time (a tendency not to rapidly disorganize) is needed. The point is just that given that certain preconditions are met, the same functionality can be pressed from multiple different materials and designs. For some famous opposition to this view, see Searle (1980, 1992).

What exactly is meant by casting the brain as a machine, albeit one made out of meat? There exists a historical trend, to be sure, of trying to understand the workings of the brain by analogy with various currently fashionable technologies: the telegraph, the steam engine, and the telephone switchboard are all said to have had their day in the sun. But the "meat machine" phrase is intended, it should now be clear, to do more than hint at some rough analogy. For with regard to the very special class of machines known as computers, the claim is that the brain (and, by not unproblematic extension, the mind) actually *is* some such device. It is not that the brain is somehow *like* a computer: everything is like everything else in some respect or other. It is that neural tissues, synapses, cell assemblies, and all the rest are just nature's rather wet and sticky way of building a hunk of honest-to-God computing machinery. Mindware, it is then claimed, is found "in" the brain in just the way that software is found "in" the computing system that is running it.

The attractions of such a view can hardly be overstated. It makes the mental special without making it ghostly. It makes the mental depend on the physical, but in a rather complex and (as we shall see) liberating way. And it provides a ready-made answer to a profound puzzle: how to get sensible, reason-respecting behavior

out of a hunk of physical matter. To flesh out this idea of nonmysterious reason-respecting behavior, we next review some crucial developments[1] in the history (and prehistory) of artificial intelligence.

One key development was the appreciation of the power and scope of formal logics. A decent historical account of this development would take us too far afield, touching perhaps on the pioneering efforts in the seventeenth century by Pascal and Leibniz, as well as on the twentieth-century contributions of Boole, Frege, Russell, Whitehead, and others. A useful historical account can be found in Glymour, Ford, and Hayes (1995). The idea that shines through the history, however, is the idea of finding and describing "laws of reason"—an idea whose clearest expression emerged first in the arena of formal logics. Formal logics are systems comprising sets of symbols, ways of joining the symbols so as to express complex propositions, and rules specifying how to legally derive new symbol complexes from old ones. The beauty of formal logics is that the steadfast application of the rules guarantees that you will never legally infer a false conclusion from true premises, even if you have no idea what, if anything, the strings of symbols actually mean. Just follow the rules and truth will be preserved. The situation is thus a little (just a little) like a person, incompetent in practical matters, who is nonetheless able to successfully build a cabinet or bookshelf by following written instructions for the manipulation of a set of preprovided pieces. Such building behavior can look as if it is rooted in a deep appreciation of the principles and laws of woodworking: but in fact, the person is just blindly making the moves allowed or dictated by the instruction set.

Formal logics show us how to preserve at least one kind of semantic (meaning-involving: see Box 1.2) property without relying on anyone's actually appreciating the meanings (if any) of the symbol strings involved. The seemingly ghostly and ephemeral world of meanings and logical implications is respected, and in a certain sense recreated, in a realm whose operating procedures do not rely on meanings at all! It is recreated as a realm of marks or "tokens," recognized by their physical ("syntactic") characteristics alone and manipulated according to rules that refer only to those physical characteristics (characteristics such as the shape of the symbol—see Box 1.2). As Newell and Simon comment:

> Logic . . . was a game played with meaningless tokens according to certain purely syntactic rules. Thus progress was first made by walking away from all that seemed relevant to meaning and human symbols. (Newell and Simon, 1976, p. 43)

Or, to put it in the more famous words of the philosopher John Haugeland:

> If you take care of the syntax, *the semantics will take care of itself.* (Haugeland, 1981a, p. 23, original emphasis)

[1]The next few paragraphs draw on Newell and Simon's (1976) discussion of the development of the Physical Symbol Hypothesis (see Chapter 2 following), on John Haugeland's (1981a), and on Glymour, Ford, and Hayes' (1995).

Box 1.2

Syntax and Semantics

Semantic properties are the "meaning-involving" properties of words, sentences, and internal representations. *Syntactic* properties, at least as philosophers tend to use the term, are nonsemantic properties of, for example, written or spoken words; or of any kinds of inscriptions of meaningful items (e.g., the physical states that the pocket calculator uses to store a number in memory). Two synonymous written words ("dog" and "chien") are thus semantically identical but syntactically distinct, whereas ambiguous words ("bank" as in river or "bank" as in Main Street) are syntactically identical but semantically distinct. The idea of a *token* is the idea of a specific syntactic item (e.g., *this* occurrence of the word "dog"). A pocket calculator manipulates physical tokens (inner syntactic states) to which the operation of the device is sensitive. It is by being sensitive to the distinct syntactic features of the inner tokens that the calculator manages to behave in an arithmetic-respecting fashion: it is set up *precisely* so that syntax-driven operations on inner tokens standing for numbers respect meaningful arithmetical relations between the numbers. Taking care of the syntax, in Haugeland's famous phrase, thus allows the semantics to take care of itself.

This shift from meaning to form (from semantics to syntax if you will) also begins to suggest an attractive liberalism concerning actual physical structure. For what matters, as far as the identity of these formal systems is concerned, is not, for example, the precise shape of the symbol for "and." The shape could be "AND" or "and" or "&" or "^" or whatever. All that matters is that the shape is used consistently and that the rules are set up so as to specify how to treat strings of symbols joined by that shape: to allow, for example, the derivation of "A" from the string "A and B." Logics are thus first-rate examples of *formal systems* in the sense of Haugeland (1981a, 1997). They are systems whose essence lies not in the precise physical details but in the web of legal moves and transitions.

Most games, Haugeland notes, are formal systems in exactly this sense. You can play chess on a board of wood or marble, using pieces shaped like animals, movie stars, or the crew of the starship Enterprise. You could even, Haugeland suggests, play chess using helicopters as pieces and a grid of helipads on top of tall buildings as the board. All that matters is again the web of legal moves and the physical distinguishability of the tokens.

Thinking about formal systems thus liberates us in two very powerful ways at a single stroke. Semantic relations (such as truth preservation: if "A and B" is true, "A" is true) are seen to be respected in virtue of procedures that make no intrinsic reference to meanings. And the specific physical details of any such system are seen to be unimportant, since what matters is the golden web of moves and transitions.

Semantics is thus made unmysterious without making it brute physical. Who says you can't have your cake and eat it?

The next big development was the formalization (Turing, 1936) of the notion of computation itself. Turing's work, which predates the development of the digital computer, introduced the foundational notion of (what has since come to be known as) the Turing machine. This is an imaginary device consisting of an infinite tape, a simple processor (a "finite state machine"), and a read/write head. The tape acts as a data store, using some fixed set of symbols. The read/write head can read a symbol off the tape, move itself one square backward or forward on the tape, and write onto the tape. The finite state machine (a kind of central processor) has enough memory to recall what symbol was just read and what state it (the finite state machine) was in. These two facts together determine the next action, which is carried out by the read/write head, and determine also the next state of the finite state machine. What Turing showed was that some such device, performing a sequence of simple computations governed by the symbols on the tape, could compute the answer to any sufficiently well-specified problem (see Box 1.3).

We thus confront a quite marvelous confluence of ideas. Turing's work clearly suggested the notion of a physical machine whose syntax-following properties would enable it to solve any well-specified problem. Set alongside the earlier work on logics and formal systems, this amounted to nothing less than

> . . . the emergence of a new level of analysis, independent of physics yet mechanistic in spirit . . . a science of structure and function divorced from material substance. (Pylyshyn, 1986, p. 68)

Thus was classical cognitive science conceived. The vision finally became flesh, however, only because of a third (and final) innovation: the actual construction of general purpose electronic computing machinery and the development of flexible, high-level programming techniques. The bedrock machinery (the digital computer) was designed by John von Neumann in the 1940s and with its advent, all the pieces seemed to fall finally into place. For it was now clear that once realized in the physical medium of an electronic computer, a formal system could run *on its own,* without a human being sitting there deciding how and when to apply the rules to initiate the legal transformations. The well-programmed electronic computer, as John Haugeland nicely points out, is really just an automatic ("self-moving") formal system:

> It is like a chess set that sits there and plays chess by itself, without any intervention from the players, or an automatic formal system that writes out its own proofs and theorems without any help from the mathematician. (Haugeland, 1981a, p. 10; also Haugeland, 1997, pp. 11–12)

Of course, the machine needs a program. And programs were, in those days (but see Chapter 4), written by good old fashioned human beings. But once the program was in place, and the power on, the machine took care of the rest. The transitions between legal syntactic states (states that also, under interpretation, *meant* something) no longer required a human operator. The physical world

Box 1.3

A Turing Machine

To make the idea of Turing machine computation concrete, let us borrow an example from Kim (1996, pp. 80–85). Suppose the goal is to get a Turing machine to add positive numbers. Express the numbers to be added as a sequence of the symbols "#" (marking the beginning and end of numbers) "1" and "+," So the sum 3 + 2 is encoded on the tape as shown in Figure 1.1. A neat program for adding the numbers (where "A" indicates the initial location and initial state of the read/write head) is as follows:

Instruction 1: If read-write head is in machine state A and encounters a "1," it moves one square to the right, and the head stays in state A.
Instruction 2: If the head is in state A and encounters a "+," it replaces it with a "1," stays in state A, and moves one square to the right.
Instruction 3: If the head is in state A and it encounters a "#," move one square left and go into machine state B.
Instruction 4: If the head is in machine state B and encounters a "1,"delete it, replace with a "#," and halt.

You should be able to see how this works. Basically, the machine starts "pointed" at the leftmost "1." It scans right seeking a "+," which it replaces with a "1." It continues scanning right until the "#" indicates the end of the sum, at which point it moves one square left, deletes a single "1," and replaces it with a "#." The tape now displays the answer to the addition problem in the same notation used to encode the question, as shown in Figure 1.2.

Similar set-ups (try to imagine how they work) can do subtraction, multiplication, and more (see Kim, 1996, pp. 83–85). But Turing's most striking achievement in this area was to show that you could then define special kind of Turing machine (the aptly-named universal Turning machine) able to imitate any other Turing machine. The symbols on the tape, in this universal case, encode a description of the behavior of the other machine. The universal Turning machine uses this description to mimic the input–output function of any other such device and hence is itself capable of carrying out *any* sufficiently well-specified computation. (For detailed accounts see Franklin, 1995; Haugeland, 1985; Turing, 1936, 1950.)

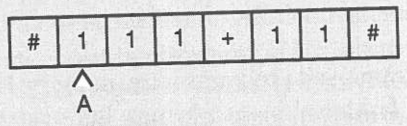

Figure 1.1 (After Kim, 1996, p. 81)

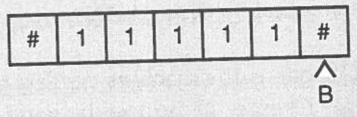

Figure 1.2 (After Kim, 1996, p. 81)

The Turing machine affords a fine example of a simple case in which syntax-driven operations support a semantics-respecting (meaning-respecting) process. Notice also that you could *build* a simple Turing machine out of many different materials. It is the formal (syntactic) organization that matters for its semantic success. There is a useful little interactive JavaScipt Turing Machine simulator available online, at http://www.turing.org.uk/turing/scrapbook/tmjava.html

suddenly included clear, nonevolved, nonorganic examples of what Daniel Dennett would later dub "syntactic engines"—quasi-autonomous systems whose sheer physical makeup ensured (under interpretation) some kind of ongoing reason-respecting behavior. No wonder the early researchers were jubilant! Newell and Simon nicely capture the mood:

> It is not my aim to surprise or shock you. . . . But the simplest way I can summarize is to say that there are now in the world machines that think, that learn and that create. Moreover, their ability to do these things is going to increase rapidly until—in a visible future—the range of problems they can handle will be co-extensive with the range to which the human mind has been applied. (Newell and Simon, 1958, p. 6; quoted in Dreyfus and Dreyfus, 1990, p. 312)

This jubilant mood deepened as advanced programming techniques[2] brought forth impressive problem-solving displays, while the broader theoretical and philosophical implications (see Box 1.4) of these early successes could hardly have been more striking. The once-mysterious realm of mindware (represented, admittedly, by just two of its many denizens: truth preservation and abstract problem solving) looked ripe for conquest and understanding. Mind was not ghostly stuff, but the operation of a formal computational system implemented in the meatware of the brain.

Such is the heart of the matter. Mindware, it was claimed, is to the neural meat machine as software is to the computer. The brain may be the standard (local, earthly, biological) implementation—but cognition is a program-level thing. Mind is thus ghostly enough to float fairly free of the gory neuroscientific details. But it is not so ghostly as to escape the nets of more abstract (formal, computational) scientific investigation. This is an appealing story. But is it correct? Let's worry.

[2]For example, list-processing languages, as pioneered in Newell and Simon's Logic Theorist program in 1956 and perfected in McCarthy's LISP around 1960, encouraged the use of more complex "recursive programming" strategies in which symbols point to data structures that contain symbols pointing to further data structures and so on. They also made full use of the fact that the same electronic memory could store both program and data, a feature that allowed programs to be modified and operated on in the same ways as data. LISP even boasted a universal function, EVAL, that made it as powerful, modulo finite memory limitations, as a universal Turing machine.

Box 1.4

MACHINE FUNCTIONALISM

The leading philosophical offspring of the developments in artificial intelligence went by the name of *machine functionalism*, and it was offered as an answer to one of the deepest questions ever asked by humankind; namely, what is the essence (the deep nature) of the mental? What fundamental facts make it the case that some parts of the physical world have mental lives (thoughts, beliefs, feelings, and all the rest) and others do not? Substance dualists, recall, thought that the answer lay in the presence or absence of a special kind of mental *stuff*. Reacting against this idea (and against so-called philosophical behaviorism—see Appendix I). Mind-brain identity theorists, such as Smart (1959) (and again, see Appendix I), claimed that mental states *just are* processes going on in the brain. This bald identity claim, however, threatened to make the link between mental states and specific, material brain states a little too intimate. A key worry (e.g., Putnam, 1960, 1967) was that if it was really essential to being in a certain mental state that one be in a specific brain state, it would seem to follow that creatures lacking brains built just like ours (say, Martians or silicon-based robots) could not be in those very same mental states. But surely, the intuition went, creatures with very different brains from ours could, at least in principle, share, for example, the belief that it is raining. Where, then, should we look for the commonality that could unite the robot, the Martian, and the Bostonian? The work in logic and formal systems, Turing machines, and electronic computation now suggested an answer: look not to the specific physical story (of neurons and wetware), nor to the surface behavior, but to the inner organization; that is to say, to the golden web: to the abstract, formal organization of the system. It is this organization—depicted by the machine functionalists as a web of links between possible inputs, inner computational states, and outputs (actions, speech)—that fixes the shape and contents of a mental life. The building materials do not matter: the web of transitions could be realized in flesh, silicon, or cream cheese (Putnam, 1975b, p. 291). To be in such and such a mental state is simply to be a physical device, of whatever composition, that satisfies a specific formal description. Mindware, in humans, happens to run on a meat machine. But the very same mindware (as picked out by the web of legal state transitions) might run in some silicon device or in the alien organic matter of a Martian.

1.2 Discussion

(A brief note of reassurance: many of the topics treated below recur again and again in subsequent chapters. At this point, we lack much of the detailed background needed to really do them justice. But it is time to test the waters.)

(A) WHY TREAT THOUGHT AS COMPUTATION?

Why treat thought as computation? The principal reason (apart from the fact that it seems to work!) is that thinkers are physical devices whose behavior patterns are reason respecting. Thinkers act in ways that are usefully understood as sensitively guided by reasons, ideas, and beliefs. Electronic computing devices show us one way in which this strange "dual profile" (of physical substance and reason-respecting behavior) can actually come about.

The notion of reason-respecting behavior, however, bears immediate amplification. A nice example of this kind of behavior is given by Zenon Pylyshyn. Pylyshyn (1986) describes the case of the pedestrian who witnesses a car crash, runs to a telephone, and punches out 911. We could, as Pylyshyn notes, try to explain this behavior by telling a purely physical story (maybe involving specific neurons, or even quantum events, whatever). But such a story, Pylyshyn argues, will not help us understand the behavior in its *reason-guided* aspects. For example, suppose we ask: what would happen if the phone was dead, or if it was a dial phone instead of a touch-tone phone, or if the accident occurred in England instead of the United States? The neural story underlying the behavioral response will differ widely if the agent dials 999 (the emergency code in England) and not 911, or must run to find a working phone. Yet commonsense psychological talk makes sense of all these options at a stroke by depicting the agent as seeing a crash and *wanting to get help*. What we need, Pylyshyn powerfully suggests, is a scientific story that remains in touch with this more abstract and reason-involving characterization. And the simplest way to provide one is to imagine that the agent's brain contains states ("symbols") that represent the event *as* a car crash and that the computational state-transitions occurring inside the system (realized as physical events in the brain) then lead to new sets of states (more symbols) whose proper interpretation is, for example, "seek help," "find a telephone," and so on. The interpretations thus glue inner states to sensible real-world behaviors. Cognizers, it is claimed, "instantiate . . . representation physically as cognitive codes and . . . their behavior is a causal consequence of operations carried out on those codes" (Pylyshyn, 1986, p. xiii).

The same argument can be found in, for example, Fodor (1987), couched as a point about content-determined transitions in trains of thought, as when the thought "it is raining" leads to the thought "let's go indoors." This, for Fodor (but see Chapters 4 onward), is the essence of human rationality. How is such rationality mechanically possible? A good empirical hypothesis, Fodor suggests, is that there are neural symbols (inner states apt for interpretation) that mean, for example, "it is raining" and whose physical properties lead in context to the generation of other symbols that mean "let's go indoors." If that is how the brain works then the brain is indeed a computer in exactly the sense displayed earlier. And if such were the case, then the mystery concerning reason-guided (content-determined) transitions in thought is resolved:

> If the mind is a sort of computer, we begin to see how . . . there could be non-arbitrary content-relations among causally related thoughts. (Fodor, 1987, p. 19)