6 Functionalism

6.1 The Rise of Functionalism

The identity theory enjoyed a surprisingly brief period of favor among philosophers. Its decline in popularity was not the result of dualist counter-attacks, however, but a consequence of the rise of a very different conception of mind: functionalism. Functionalists were not put off by identity theorists’ commitment to materialism. Although, as you will see, functionalism is not a materialist theory per se, functionalism can be understood as compatible with the aims of materialism; most functionalists would regard themselves as materialists of one sort or another.

Functionalists would allow that, although immaterial substances – Cartesian souls, for instance – are conceivable, in all probability every substance is a material substance. Were that so, every property possessed by a substance would be possessed by a material substance. Does this imply that every property is a material property? Are mental properties a species of material property? The issues here are tricky. They occupy the sections – and chapters – that follow.

These days functionalism dominates the landscape in the philosophy of mind, in cognitive science, and in psychology. Functionalism offers a perspective on the mind that suits the needs of many empirical scientists, a perspective that promises solutions to a host of long-standing philosophical puzzles about minds and their relation to material bodies. Clearly functionalism – the doctrine, if not the label – has etched its way into the popular imagination by way of the press and television. When basic tenets of functionalism are put to non-philosophers, the response is, often enough, ‘Well, that’s obvious, isn’t it?’

This is not to say that functionalism lacks critics. On the contrary, plenty of philosophers and empirical scientists have found functionalism wanting. There is scant agreement among its opponents, however, concerning where exactly functionalism falls down. Indeed, opponents are typically willing to concede that functionalism is right about some things – although, again, what these things are is something concerning which there is little consensus. In the absence of clear competitors, many theorists have opted to stick with functionalism despite what they admit are gaps and deficiencies, at least until something better comes along. In this respect, functionalism wins by default.

6.2 The Functionalist Picture

Functionalism’s emergence coincided with the meteoric rise of interest in computation and computing machines in the 1950s and 1960s. When you consider computational operations in a computing machine, you ‘abstract’ from the machine’s hardware. Two very differently constructed mechanisms can be said to perform the same computations, run the same programs. Charles Babbage (1792–1871) is usually credited with the design of the first programmable computing machine. Although the history is murky, Augusta Ada King (1815–52), daughter of Lord Byron and Countess of Lovelace, appears to have recognized the computational potential of Babbage’s machine and formulated an algorithm for calculating a sequence of Bernoulli numbers, making her the first ‘computer programmer’.

Babbage’s design called for a device made of brass gears, cylinders, rods, levers, and assorted mechanical gizmos. Fully assembled, this mechanical marvel – Babbage christened it the Analytical Engine – would have been the size of a railway locomotive. Although the machine was never completed, had it been, there is no reason to doubt that it could have performed (rather more slowly) the very sorts of computation that twenty-first century computers perform. Where Babbage used gears and cylinders, the first electronic computing machines, constructed in the 1950s and early 1960s, made use of vacuum tubes. Today’s computing machines have replaced vacuum tubes with arrays of millions of minuscule transistors embedded in slivers of silicon.

Economies of scale result when you move from brass gears and cylinders to vacuum tubes, and again when you move from vacuum tubes to densely packed transistors. These economies make an enormous practical difference in the range of computations you could expect a given device to perform. When you consider only the computations themselves, however, all such devices are on a par. One might be faster, or more reliable, or less expensive to manufacture than another, but all carry out the same kinds of computation. For this reason, when you consider computations – the manipulation of symbols in accord with formal rules – you ‘abstract’ from the material nature of the device performing them. And in so doing, you characterize the behavior of computing devices at a ‘higher level’.

Are computational processes material processes? Philosophers of a functionalist bent prefer to say that computational processes are implemented by or ‘realized’ in material systems. The material process that realizes a given computational operation in Babbage’s Analytical Engine differs from material processes that realize it in a modern transistor-based computing machine or in an antique device equipped with vacuum tubes. If there are immaterial
substances, the very same process could well have an immaterial realization. Functionalists sum up these points by describing computational processes as ‘multiply realizable’.

Think of a computing machine as a device that operates in a way that could be described by reference to, and whose behavior could be explained in terms of, formal operations over symbols: symbol manipulation, for short. Such a device could be made of any number of materials – or even, perhaps, of immaterial ghost-stuff – and organized in any number of ways. In considering a device as a computational device, you consider it without concern for its actual makeup. Just as you ‘abstract’ from the size, color, and spatial location of a geometrical figure when it is the subject of a geometrical proof, so you ‘abstract’ from a computing machine’s material composition when you consider it as a computational device.

### 6.3 Abstraction as Partial Consideration

The notion of ‘abstraction’ in play here is worth making explicit. Imagine that you are watching Lilian run through a proof of the Pythagorean theorem. Lilian draws a right triangle on a sheet of paper with a red crayon. You attend to the shape of the drawing, not its color, size, or spatial orientation. You attend to the lines and not their wiggliness. In so doing you ‘abstract’ from the drawing’s color, size, and spatial orientation. You engage in what Locke felicitously described as ‘partial consideration’. Human beings (and doubtless other intelligent creatures) have a capacity for this kind of selective attention. You can attend to or consider the color of a paint chip without attending to or considering its size or shape. You might do this when, for instance, you are deciding on what color to paint your bedroom.

In thinking of a device as performing computations, you are abstracting in this sense. You are considering the device as a finite symbol processor, not as something made of metal and plastic that operates in accord with laws of physics and chemistry. In describing computational or functional systems as ‘abstract’, you are not describing nonmaterial abstract entities. You are simply describing systems without reference to their material characteristics. In the same way, in describing Lilian’s right triangle as a triangle, you are ‘abstracting’ from the triangle’s having wiggly lines or being red, you are not describing a nonmaterial triangle existing alongside or in addition to the red, right triangle.

I am emphasizing this topic because philosophers sometimes leave students with the impression that abstraction is a matter of identifying mysterious abstract entities taken to subsist in a realm of abstracta, distinct from the realm of ‘concrete’ material entities. Think of ‘abstracting’ as a matter of omitting, leaving out of consideration. An abstract description leaves out features of the object described: to abstract is to engage in partial consideration. You can accept talk of ‘abstraction’ in this sense without imagining that such talk invokes a realm of spectral entities.

### 6.4 Minds as Programs

Suppose you thought of minds in roughly the way you might think of computing machines. A mind is a device capable of performing particular sorts of operation. States of mind resemble computational states, at least to the extent that they could occur, in principle, in many different kinds of material (and perhaps immaterial, a qualification I shall henceforth omit) system. To talk of minds and mental operations, is to abstract from whatever ‘realizes’ them: to talk at a ‘higher level’.

This preliminary characterization is intended only to impart the flavor of functionalism. You should not be put off by the idea that creatures like us, creatures possessing minds, are ‘nothing more than machines’. The point of the computing machine analogy is not to suggest that you are a mechanical automaton, a robot rigidly programmed to behave as you do. The point, rather, is that you might think of minds as bearing a relation to their material embodiments analogous to the relation computer programs bear to devices on which they run. A program is ‘embodied’ in some material device or other. But the very same program could be compiled to run on very different sorts of material device. In the same vein, you might suppose that every mind has some material embodiment, although minds could be embodied in very different kinds of material. In the case of human beings, our brains constitute the hardware on which our mental software runs. Alpha Centaurians, in contrast, might share our psychology, our mental software, yet have very different, perhaps non-carbon-based, hardware.

If something like this were right, then there would seem to be no deep mystery as to how minds and bodies are related. Minds are not identifiable with brains, as an identity theorist would have it; but neither are minds distinct immaterial substances mysteriously linked to bodies. Talk of minds is at bottom talk of material systems at a ‘higher level’, a level that abstracts from the ‘hardware’. Feeling a pain or thinking of Vienna are not brain processes, any more than a computational operation, summing two integers for instance, is a vacuum tube process or a transistor process. Brains and brain processes are analogous to the hardware and electrical processes that, in a particular kind of computing machine, realize computations. Thoughts and feelings are, as computations are, multiply realizable. They are capable of being embodied in a potentially endless array of organisms or devices. Your mind is a program running in your brain.
6.13 Moving Beyond Analogy

In introducing functionalism, I appealed to a computing machine analogy: minds are analogous to software running on computing machines. Mental states are realized by states of the nervous systems of intelligent creatures in the way computational states are realized by hardware states. Suppose the analogy is apt, suppose mental states are analogous to computational states. The question now arises, how do minds actually work? What are the principles of operation of mental mechanisms?

One possibility – some would say the only possibility that makes sense – is that mental states are not merely analogous to computational states; mental states are a species of computational state. We understand computation. You can see how a computer programmed in a particular way does what it does. When you ask how minds do what they do, the possibility that minds are computational devices immediately suggests itself. And really, if mental processes are not computational processes, what are they?

Indeed, once you entertain the thought that mental processes could be computational processes, the possibility might strike you as overwhelmingly obvious. Computing machines are fed information, process it, and perform various tasks in light of information stored in memory. There is no mystery as to how a computing machine could manipulate symbols in accord with principles incorporated in its program. What else could thinking be if not the processing of symbols in a principled way? Perhaps the computing machine analogy is more than an analogy, perhaps it is a recipe for understanding the nature of the mind. This is the thought to be explored in Chapter 7.

Suggested Reading

Although versions of functionalism have been with us since Aristotle (see Nussbaum and Rorty’s Essays on Aristotle’s De Anima, 1992), the current wave of functionalism could be said to have begun with Hilary Putnam’s ‘Minds and Machines’ (1960). Putnam’s subsequent ‘Psychological Predicates’ (1967) spelled the doctrine out explicitly. Most readers are familiar with this paper under a different title: ‘The Nature of Mental States’. This was the title Putnam used when the paper was reprinted, a decision pregnant with philosophical implications (or so I argue in §§11.8, 12.12–12.13).

7 The Representational Theory of Mind

7.1 Mental Representation

Functionalism affords a highly abstract conception of the mind. One especially influential elaboration of this conception depicts minds as symbol processors. To see how this might work, imagine a schematic specification of a functional system resembling the flow chart model deployed in the characterization of organizational hierarchies and computer programs (recall Figures 6.2 and 6.5). Figure 7.1 provides a rough idea of what such a flow chart might look like.

To keep matters simple in the discussion to follow, I propose to focus just on beliefs and desires, omitting the system’s other elements. Now, rather than conceiving of beliefs and desires individually, think of the mind as including a ‘belief box’ and a ‘desire box’ (Figure 7.2). Occupants of the belief box function as beliefs, occupants of the desire box function as desires. But what are these occupants?

Consider your belief that the window is open. More particularly, consider the content of this belief: that the window is open. A belief with this content can be distinguished from beliefs that have different contents; your belief, for instance, that the room is chilly. You could think of a belief as having two ‘dimensions’. First, there is a ‘propositional content’, typically expressed in English via a that-clause: that the window is open, that the room is chilly, that snow is white. Second, there is an attitudinal component: in believing that the window is open, you have a particular attitude, an attitude of acceptance, toward a particular proposition. Compare believing that the window is open, and desiring that the window is open: same contents, different attitudes.

Philosophers call beliefs and desires – together with intentions, wishes, hopes, and assorted other ‘contentful’ states of mind – propositional attitudes. In harboring a belief or a desire, you take up an attitude toward a proposition – that the window is open, for instance.

What are propositions? Propositions are abstractions (see § 6.3) When you take an ordinary sentence, ‘Snow is white’, and consider just its meaning, you are considering the proposition the sentence expresses. You could think of propositions as meanings. When you utter a sentence, you express a particular proposition. As an English speaker, you can say that ‘Snow is white’ means that snow is white. Here, the sentence and what it means align. But you would not get far without distinguishing sentences and their meanings, distinguishing sentences and propositions those sentences express.

First, distinct sentences in the same language can have the same meaning, can express the same proposition. Thus, the sentences

1. Bachelors procrastinate
2. Unmarried adult males put things off

evidently have the same meaning. Think of the common meaning as a proposition: the two sentences express the same proposition. So, even within a language, there is no simple, one–one mapping between sentences and meanings. (I leave it to the reader to think of examples in which a single sentence could be used to mean different things, to express different propositions.)

Second, and more importantly in the present context, sentences in distinct languages can have the same meaning. The sentences

3. It’s raining
4. Il pleut
5. Es regnet
mean the same in English, French, and German, respectively. Each of these sentences could then be said to express the same proposition, the proposition that it is raining. Talk of propositions provides a way to talk about meanings independently of sentences used to express those meanings.

Although philosophers regard propositions as indispensable abstractions, we need sentences to express – or, for that matter, to discuss – propositions. Sentences are, so to speak, concrete, here and now propositional vehicles. Sentences stand in for propositions in the material world. This simple point takes on added significance when you start thinking about how the mind works. Thoughts are meaningful. Thoughts express propositions. Beliefs, desires, intentions and the like can be understood as attitudes toward propositions. How might this central feature of our mental lives be fitted into a functionalist framework?

Here is one possibility. Think about states of mind as something like strings of symbols, sentences. Sentences can occupy one or another ‘box’ in a functional system. (Indeed instances of a symbolic string – a sentence – could appear simultaneously in more than one box: you could believe and desire that the window is open.) Sentences can be moved into and out of particular boxes in response to what occurs elsewhere in the system.

The idea here is that your forming a belief that the window is open would be a matter of constructing a symbol expressing the proposition that the window is open and inserting it into your ‘belief box’. In the same vein, your wanting the window to be open would be your placing such a symbol in your ‘desire box’. Your belief box and your desire box are connected in distinctive ways to the rest of the system constituting your mind. If a symbol representing the proposition that the window is closed is in your desire box, for instance, this might – in conjunction with the presence of various other symbols in your belief and desire boxes – lead you to walk across the room and lower the window (Figure 7.3). The presence of the same symbol in your belief box (assuming that it is absent from your desire box) might – and, again, in conjunction with the presence of other symbols in your belief and desire boxes – lead to markedly different behavior.

This way of thinking of functional systems enables us to see more clearly how creatures with very different beliefs and desires might nevertheless be seen as functionally on a par. You, an infant, and Richard Feynman all fit the highly simplified models in Figures 7.1 and 7.2. You differ, however, with respect to the symbols that are apt to appear in your respective belief and desire boxes.

This conception of mind, the ‘Representational Theory of Mind’, has been enthusiastically defended by Jerry Fodor since the 1970s. The Representational Theory of Mind requires the postulation of a system of symbols that function as ‘mental representations’. These symbols make up what Fodor calls a ‘Language of Thought’, a biologically fixed ‘code’ analogous to the ‘machine code’ hard-wired into an ordinary computing machine. Your forming a belief that the window is open is a matter of a sentence in the Language of Thought corresponding to the English sentence, ‘The window is open’, a sentence expressing the proposition that the window is open acquiring an appropriate functional role, a matter of this sentence’s being moved into your belief box.

On this model, mental operations are taken to be ‘computations over symbols’. Talk of belief and desire ‘boxes’ is to be understood as shorthand for talk of specific kinds of computation. If you thought of the mind as a kind of program running on biological ‘hardware’, belief and desire boxes would represent subroutines that processes symbols in distinctive ways.

### 7.2 Semantic Engines

Fodor and his allies have long insisted that the Representational Theory of Mind (and with it the Language of Thought hypothesis) is the only game in town. The Representational Theory of Mind provides a way of understanding how minds, higher-level entities, could systematically affect and be affected by bodily goings-on. Until someone produces a serious competitor, they argue, the theory wins by default.

But how is all this supposed to work? What could it mean to speak of sentences in a Language of Thought shunting into and out of belief and desire boxes? Notice first, that the focus here is on sentence tokens. A sentence token – a particular ‘inscription’ – is a concrete entity, something that could exert causal influence. A sentence token is to be distinguished from a sentence type.

To appreciate the distinction, consider the sentences in the box below (Figure 7.4).

How many sentences does the box contain? The box contains two instances, cases, or tokens of a single sentence type. The answer to the ‘how many?’
question depends on how you are counting. If you are counting sentence tokens, the answer is two: the box contains two sentences. If you are counting sentence types, the box contains just one sentence, one sentence type. How many letters are in a Scrabble set? The answer is 26 or 98, depending on whether you are counting letter types or letter tokens. When proponents of the Representational Theory of Mind speak of sentences occupying belief boxes, or sentences affecting causal processes, they mean to be speaking of sentence tokens, not types: individual entities, not kinds or types of entity.

Because functionalism and the Representational Theory of Mind describe mental operations at a high level of abstraction, it is easy to imagine that you understand them better than you do. It is one thing to describe in a very general way how a system might work, another matter to understand how this description might be implemented in concrete cases. This is so for accounts of physical processes framed in terms of ‘information’. And it is so for functionalism and the Representational Theory of Mind. The question is, how is the theory ‘implemented’?

Consider the sentences – the token sentences – printed on this page. Each one is the result of an ordinary material causal process; and each one produces ordinary material effects: it reflects light in a particular way, for instance, so as to be visible. To see why this is significant, return to the distinction between sentences and propositions sentences are used to express. When you encounter sentences in your native language, their meanings, the propositions they express, leap out at you. But of course, you can encounter sentences without having any sense of what they mean. This happens whenever you confront sentences in a language you do not understand: you see or hear only marks on paper or sounds.

Now, suppose you set out to build a device capable of manipulating meaningful sentences. How would you begin? When you read a sentence you grasp its meaning and respond to that meaning. But what steps would you take to endow the device with a capacity to ‘grasp meanings’, to understand? Opting to include a ‘meaning grasping’ module, would be to invite the question, how do you do that? And this is the original question all over again.

Suppose you constructed a device that:

1. Operated on well-understood mechanical principles, principles the implementation of which poses no insurmountable technical difficulties, but that
2. Its operation was apparently ‘intelligent’.

Imagine, for instance, a device that manipulated sentences without regard to their meanings, but did so in a way that coincided with the way those sentences would be manipulated by an agent who grasped their meanings. Such a device – what John Haugeland dubs a ‘semantic engine’ – would perfectly mimic the performance of a native speaker, but would do so without relying, as a native speaker would, on a grasp of the meanings of sentences it manipulated. Those sentences might express propositions, but the device would ‘care’ only about their shapes, their ‘syntax’. (Syntax concerns only structural or ‘formal’ features of sentences; semantics concerns their meanings.) Such a device, a semantic engine, operates exclusively on syntactic principles and ‘formal’ relations among sentences, relations definable wholly by reference to sentences’ syntactic characteristics.

Yes, but is such a device possible?

Not only are semantic engines possible, they exist already, and in large numbers! An ordinary computing machine is a semantic engine. Engineers design and program computing machines so that they manipulate symbols in accord with purely syntactic, purely formal principles. The symbols are meaningful – to us – but the machines that deploy them care nothing about this. They operate on uninterpreted symbols, but – and this is the key point – in a manner that honors semantic constraints. (This is just a fancy way of saying that computing machines manipulate symbols in a way that makes sense – to us – in light of their meanings.)

How is this possible? How could syntax ‘mirror semantics’? If you have ever studied logic, you have already encountered an important example of systems that make use of purely syntactic or formal principles in the manipulation of symbols, but in a way that honors semantic relations among those symbols. Ordinary rules of inference refer only to shapes of symbols. Take the rule commonly known as modus ponens:

\[
p \supset q
\]

\[
p
\]

\[
q
\]

(In English: ‘If \( p \) then \( q \); plus \( p \), yields \( q \).)

The rule says, in effect, that if you have a particular configuration of symbols (here, \( p \supset q \) and \( p \)), you are permitted to write a new symbol (in this case, \( q \)). (Think of the \( \supset \) as expressing an English conditional – ‘if … then …’ – construction.) In formulating the rule, I have used variables \( p \) and \( q \) that range over sentences. The rule, in effect, says that whenever you have a \( \supset \) flanked by sentences, together with a sentence that matches the sentence on the left of the \( \supset \), you are permitted to ‘detach’ the sentence to the right of the \( \supset \).

For present purposes, what is significant about the modus ponens rule is that it is formulated and deployed without regard to semantics, without regard to the meanings of sentences to which it applies. Even so, applications of the rule ‘make sense’; they conform to the semantics of inference. If you accept the sentence

1. If it’s raining then I’ll need an umbrella,

and the sentence
understanding in a general way how minds could work.

**7.3 Minds as Semantic Engines**

Well and good; but what has this to do with minds? Suppose your aim were to explain the human mind by assuming that minds manipulate ‘mental representations’, and that mental representations are ‘encoded’ by sentences in the Language of Thought.

You might think that there is an obvious objection to such a project. If minds manipulate sentences, symbolic representations, then this would appear to require a sentence interpreter or understander, some component of the mind that interprets the symbols. (Recall the module posited earlier that was supposed to ‘grasp meanings’.) This would mean that you would be explaining the mind by positing within it another mind, a homunculus, a little intelligent agent whose job requires that he understand sentences in the Language of Thought and respond appropriately.

An explanation of this kind is no explanation. The point is perfectly general. You and I are watching a machine that sorts and wraps candy bars. You are impressed, and ask how the machine works. ‘Simple’, I reply. ‘There is a device inside it that controls its operations.’ Similarly, explaining that minds are capable of understanding meaningful sentences because minds include a ‘sentence understander’ is to make no progress.

Against this backdrop it is easier to appreciate the relevance of the notion of a semantic engine. Recall that a semantic engine is a device that performs symbolic operations – manipulates symbols – in a way that reflects semantic (meaningful) relations holding among these symbols, but does so exclusively by means of formal and syntactic – mechanical – principles, that is, without regard to the meanings of those symbols. Now suppose you thought that minds processed mental representations. Were minds semantic engines, you would not need to assume that minds contain components – little intelligent agents, homunculi – that understand the meanings of those representations. Ordinary computing machines are implementations of semantic engines. Perhaps brains are as well. If so, and if the brain has an appropriate functional organization, then it would seem that we have come a long way toward understanding in a general way how minds could work.

Still, you might think that there is an obvious difficulty. When scientists open up the brain, they see nothing that resembles symbols or sentences in a Language of Thought. What could it be for brains to contain and manipulate sentences?

Think of an ordinary computing machine. Engineers regard it as uncontroversial that computing machines ‘process symbols’. Yet, were you to inspect the insides of a computing machine while it is engaged in ‘symbol manipulations’, you would see nothing resembling familiar mathematical or programming symbols. Nor, incidentally, would you see any 0’s and 1’s, commonly regarded as the basic ingredients of a computing machine’s symbolic repertoire. Electronic patterns that play the role of symbols in a computing machine need not resemble your pencil-and-paper representations of those symbols. There is no reason to think that you could ‘read off’ symbols as they are manipulated by a computing machine any more than you could hope to read off a musical score by closely examining deflections in the groove of a phonograph record or the track on a compact disc. (This is not to say that you could not learn to do this. But learning to do it would require learning a complex rule that takes you from electrical events, or patterns of deflections, or magnetic patterns, to more familiar symbols.)

Thus, if the mind were a semantic engine realized by the brain, if mental operations include the manipulation of symbols, sentences in a Language of Thought, the brain’s embodiment of those symbols need not resemble the symbols you jot down using pencil and paper or type on a computer monitor. They might involve subtle electrical or chemical processes; they might be embodied in connections among neurons; they might be widely distributed in networks of simpler elements. In any case, there is no reason to imagine that such symbols could be ‘read off’ the brain in the way you read words off this page. If there is a Language of Thought, its sentences could well be indecipherable from the point of view of an observer examining the microstructure of a brain.

**7.4 The Turing Test**

The Representational Theory of Mind depicts minds as semantic engines, devices that operate on formally specifiable syntactic principles, but in a way that tracks semantic – that is, meaningful – relations among symbols, sentences in a Language of Thought. Although mental mechanisms would be indifferent to the significance of symbols on which they operate, operations performed by those mechanisms mirror operations that might be performed by an intelligent agent who understood the symbols and acted on the basis of that understanding.

When you ask a friend to print a document for you, your friend understands your request, and acts on the basis of that understanding. (How the friend responds will depend on the friend’s current beliefs and desires.) When you ‘ask’ your desktop computer to print a document by means of a typed
or voiced command, the device does not act on the basis of its grasp of the command’s meaning. The specific mechanisms that execute your command, care nothing for the command’s meaning. Even so, owing to its program, your desktop computer operates just as though it understood your command.

Now the punch line. For proponents of the Language of Thought, this is all there is to understanding. *You* can be said to understand the sentences on this page. But mental mechanisms responsible for your understanding need not *themselves* understand. On the contrary, if you thought they did, you would be hard-pressed to account for your grasp of what the sentences mean. As noted earlier, it would be a bad idea to try to explain the kind of understanding required for meaningful, intelligent thought by positing an internal understander. To avoid trivialization, an explanation must show how meaningful thought could result from the operation of mechanisms that themselves do not need to understand meanings. *Your* entertaining meaningful thoughts might thus be explained by supposing your mind includes an organized collection of components, none of which needs to understand the (meaningful) symbols it processes.

The intimate connection between semantic engines and intelligent thought lies behind an influential discussion of the possibility of ‘artificial intelligence’ published in 1950 by A. M. Turing (1912–54). Turing was a prominent mathematician whose pioneering work on the theory of computation underlies the development of modern computing machines. Turing was interested in the question whether it might be possible to build an intelligent machine. After reviewing failed attempts to define ‘intelligence’ (and, for that matter ‘machine’), Turing proposed to define intelligence ‘operationally’ and thereby to bypass altogether the vexed question of what exactly constitutes ‘real’ intelligence.

An operational definition takes the form of a test for determining whether the defined term applies. Thus, you might operationally define ‘length’ in terms of measurement procedures. Something has length if, for instance, it could be measured using a standard meter stick. Something is a meter long if its ends align with the ends of the meter stick. The Turing test is designed to ensure that whatever passed the test would qualify as intelligent. A computing machine would count as intelligent if it managed to pass the test and not otherwise.

The Turing test is based on a game Turing dubs the ‘Imitation Game’. The Imitation Game is played by three ordinary people. One, the interrogator, puts questions to two players, a man and a woman, and tries to work out which is which, which is the man, which the woman. One of the two players must answer truthfully, the other tries to mislead the interrogator. To prevent the interrogator from seeing the players or hearing their voices, the interrogator is stationed in a separate room and communicates with the players by means of a teletype (a primitive form of text messaging). By asking clever questions, the interrogator will sometimes win the game, but will sometimes lose as well. Suppose, over a large number of games, the interrogator wins – that is, correctly discovers which player is which – about half the time.

Now, says Turing, replace one member of the pair being interrogated, the man or the woman, with a computing machine programmed to take the part of the untruthful contestant. If the machine could fool a clever interrogator about as often as a human player could (as we are supposing, about half the time), the machine passes the test: the machine would count as intelligent. (If this seems too easy, you might reflect on the fact that no existing or currently contemplated computing machine comes remotely close to exhibiting the kind of resourcefulness and wit required to fool a moderately competent interrogator.)

Turing’s supposition is, in effect, that the machine in question is functioning as a semantic engine. If the machine passes the test, if the machine functions intelligently, then it would seem reasonable to say that the machine is intelligent, that the machine understands questions put to it by the interrogator, and that the machine means just what a human contestant would mean in responding as it does to the interrogator’s questions. If that is right, it would seem that the Representational Theory of Mind delivers important insights concerning the nature of the mind: minds could be programs running on neurological hardware.