6

The computational theory of mind

the only game in town

—Jerry Fodor

Science fiction is full of computers and robots that can think. HAL, the computer in Stanley Kubric's *2001: A Space Odyssey*, is perhaps the most famous example, but there are lots of others: K9 in *Doctor Who*; Data in *Startrek TNG*; and—my personal favorite—the Terminator in (you guessed it) the *Terminator*. We have become comfortable, it would seem, with the idea that a machine could have thoughts. The **computational theory of mind** takes this idea one step further. According to **computationalism** the mind is, quite literally, a computer.

In this chapter we will explore the idea that the mind is a computer. In order to understand that idea we need to be clear about what a computer is. That's the task of Section 6.2. However, before we can understand what a computer is, we first need to understand the distinction between **syntax** and **semantics**. That's the task of the next section.

6.1 Syntax and semantics

It will be helpful to have a rough-and-ready distinction between *basic symbols* and *complex symbols*. A basic symbol is one which has no meaningful parts; a complex symbol is one which is made up of two or more basic symbols. I will use English words as examples of basic symbols, and I will use English sentences as examples of complex symbols. Thus, 'Fodor' is a basic symbol whereas 'Fodor wrote *The Modularity of Mind*' is a complex symbol. (It might be argued that English words can't be basic symbols because they contain meaningful parts—the letters out of which they are assembled. For present purposes I will ignore this complication.)

The syntactic properties of a symbol are the properties which can be detected simply by examining the symbol in isolation. Here are some examples. Consider the basic symbol 'Fodor'. Just by examining that symbol you can work out that it consists of five letters in a certain arrangement. (Strictly, it consists of five letter *tokens.* For the distinction between types and tokens see Section 3.1.) You can also work out that the symbol is black on white and that it is less than three centimeters long. These are all syntactic properties of the symbol 'Fodor'. In contrast, you can't work out that the symbol 'Fodor' refers to a famous American philosopher just by examining it. Nor can you tell the name and address of the typesetter who arranged that symbol on the page. So the properties 'refers to a famous American philosopher' and 'was typeset by Bloggs of 44 Carbuncle St London' are *not* syntactic properties.

It's very common for the syntactic properties of a symbol to be called the symbol's 'shape'. That's not surprising as shape is a good example of a syntactic property. In written English it is customary to end a question with a special symbol—the question mark. That symbol has a characteristic shape—?. We can easily recognize the question mark symbol by its shape. Indeed—and this will come up again later—a *machine* can recognize the question mark symbol by its shape. So shape is a syntactic property. If you're having trouble keeping in mind what the syntactic properties are, just think of them as a symbol's shape. You won't be far wrong.

What about the semantic properties of symbols? Roughly speaking, semantic properties are properties connected with the *meaning* of a symbol. For example, the **reference** of a symbol—what it refers to—is a semantic property. To continue with our example, the basic symbol 'Fodor' refers to a famous American philosopher. The **truth value** of a symbol—whether it is true or false—is also a semantic property. Note, though, that not all symbols have a truth value. The symbol 'Fodor', for example, is neither true nor false. Whilst it is true that Fodor wrote *The Modularity of Mind* and false that Fodor wrote *Alice in Wonderland*, 'Fodor' by itself is neither true nor false.

What kinds of symbols have truth values? Some (but not all) complex symbols have truth values. The complex symbol 'Fodor wrote *The Modularity of Mind*' is true; the complex symbol 'Fodor wrote *Alice in Wonderland*' is false. Both of these complex symbols make a claim about the world. The first example claims that the world is one in which Fodor wrote *The Modularity of Mind*; the second claims that the world is one in which Fodor wrote *Alice in Wonderland.* In general, symbols have the semantic properties of truth or falsity if and only if they make a claim about the world. It is now obvious why the symbol 'Fodor' is neither true nor false. By itself, 'Fodor' makes no claim about the world.

Unlike the syntactic properties, the semantic properties *cannot* be detected by examining a symbol in isolation. Think again about the symbol 'Fodor', and assume you have no idea who (or what) Fodor is. You can stare at that symbol for as long as you like and you will never work out what it refers to. In order to know that the reference of 'Fodor' is a famous American philosopher, you have to look beyond the symbol itself. In particular, you have to work out which person is connected in the right way to that symbol. Similarly, examining the complex symbol 'Fodor wrote *Alice in Wonderland*' in isolation will not allow you to determine its truth value. You have to look at the world beyond the symbol in order to determine that 'Fodor wrote *Alice in Wonderland*' is false.

In summary, the syntactic properties of a symbol can be detected just by examining the symbol. Shape is a good example of a syntactic property. Semantic properties are connected with meaning and include properties like reference and truth value. Since you cannot determine a symbol's semantic properties by examining it in isolation, semantic properties are not syntactic properties. With the distinction between syntactic and semantic properties in place, we can turn to the question, 'What's a computer?'.

6.2 What's a computer?

In this section I'm going to describe what a computer is. I'm not going to talk about keyboards and monitors; rather, I'm going to give a very general description of computation which abstracts away from a great many of the details of real computers.

Speaking very generally, a computer has two features.

1. Computers recognize and manipulate symbols *solely on the basis of their syntactic properties*. Here's an example. I can use the 'find' function of my word processing program to locate the 'Fodor' symbols in the document I'm presently typing. My computer recognizes those symbols by their syntactic properties—perhaps my computer recognizes 'Fodor' symbols by their shape. (Electronic computers don't usually recognize symbols by their shape, but let's pretend they do.) What's especially important is that computers don't recognize symbols by their semantic properties. That's not surprising. After all, my computer has no idea at all what the symbol 'Fodor' refers to. Perhaps 'Fodor' refers to a cat; perhaps it refers to the seventh moon of the seventh planet of Alpha Centuri. My computer simply does not know. *All* my computer has to go on are the properties of symbols which can be detected in isolation; that is, *all* my computer has to go on are the syntactic properties of symbols. So computers are 'syntactic engines'—devices which recognize and manipulate symbols on the basis of their syntactic properties.
2. Whilst computers recognize and manipulate symbols solely on the basis of their syntactic properties, they can nevertheless be arranged so that the way the symbols are manipulated respects the semantic properties of those symbols. Thus it is easy to program a computer so that it begins with the following two complex symbols:

   A. All philosophers are funky.

   B. Fodor is a philosopher.

And ends up with the following complex symbol:

C. Fodor is funky.

The computer recognizes and manipulates the various symbols involved by their syntactic properties; it *does not* recognize and manipulate those symbols on the basis of their semantic properties. After all, the computer does not know who Fodor is, does not know what a philosopher is, and does not know what it is to be funky. Nevertheless, the transition from A and B to C is truth preserving: if A and B are true then C is too. So, whilst the computer is sensitive only to syntactic properties, it manages to respect semantic properties like truth value.

Notice that the transition the computer makes from A and B to C is rational in this sense: the first two complex symbols *provide evidence for* the last one. This is an example of the way that computers can be programmed to carry out at least some kinds of rational inference. We noted way back in the Introduction that the causal relations between thoughts often mirror the rational relations between those thoughts. Computers provide us with our first hint of how that might be achieved.

To sum up, a computer is a device which recognizes and manipulates symbols on the basis of their syntactic properties, but still manages to respect semantic properties like truth value.

It's worth briefly mentioning that computational states and processes are multiply realizable. That is, devices which are physically quite different can nevertheless be in the same computational states and realize the same computational processes. It happens to be convenient to realize computational states and processes in electronic circuits, but in principle they can be realized by a wide range of devices. (See Weizenbaum 1976: Ch. 2 for an explanation of how to build a computer out of toilet paper and stones!) Even amongst electronic computers there is considerable diversity. Your PC and my Mac can undertake exactly the same computations even though they are, from an engineering point of view, quite different.

In the next section I will briefly describe an incredibly simple computer. The simplicity of the computer helps make it clear that it's a syntactic engine. However, it is also clear that its symbolic manipulations respect certain semantic properties.

6.4 The computational theory of mind

According to the computational theory of mind (hereafter CTM), thoughts are complex symbols which have both syntactic and semantic properties. We have seen that complex *linguistic* symbols like 'Fodor is funky' are made up of basic symbols like 'Fodor'. Similarly, according to CTM the *thought* that Fodor is funky is a complex symbol made up of basic symbols. However, we cannot assume that the mental symbol which refers to Fodor (the philosopher) is the English word 'Fodor'. It may be that the mind has a language or code of its own. We will return to this idea in Section 6.5. For the moment we will just accept that the mental symbol which refers to Fodor is the English word 'Fodor'.

So far we have seen that, according to CTM, thoughts are complex symbols with syntactic and semantic properties. What about thinking? According to CTM, thinking involves the recognition and manipulation of thoughts purely on the basis of their syntactic properties. However, whilst thoughts are manipulated solely on the basis of their syntactic properties, those manipulations respect the semantic properties of the thoughts involved. Thus, say that I believe that

D. All philosophers are funky,

and:

E. Fodor is a philosopher.

These thoughts lead me to believe that

F. Fodor is funky.

According to CTM, the mental processor which achieves the transition from D and E to F is sensitive only to the syntactic properties of the thoughts involved. Nevertheless, the transition is truth preserving; that is, if D and E are true, so is F. Moreover, the transition is rational in the sense that the thoughts D and E provide evidence for the thought F.

Putting all this together we can say that, according to CTM, thoughts are complex symbols with syntactic and semantic properties. Thinking—the manipulation of thoughts—is achieved by processors which, whilst sensitive only to the syntactic properties of the thoughts involved, nevertheless respect their semantic properties. In other words, thinking is computation.

This is, to put it mildly, a beautiful idea. Indeed, it might even be true! We have already seen that CTM offers an account of the rationality of thought; I will now briefly sketch three further virtues of CTM.

1. We have seen that, according to CTM, thinking is **computation**. And we know that computation is possible because computers—the one on your desk or at

the library—*do exactly that.* The existence of computers therefore gives modest support to CTM since it shows that at least some physical structures are capable of performing computations. (I say *modest* support because the existence of computers does not establish that the *mind* is a computer; it only shows that computation is physically possible.)

2. We saw in Chapters 3 and 4 that mental states can be multiply realized. If mental states are computational states, then computational states must be multiply realizable. In Section 6.2 we noted that computational states and processes can indeed be multiply realized. In other words, CTM accounts for the multiple realizability of mental states.
3. CTM requires that thoughts are complex symbols. We saw in Section 6.1 that a complex symbol is built up from basic symbols. (Recall the way that the complex symbol 'Fodor wrote *The Modularity of Mind*' contains the basic symbol 'Fodor'.) More precisely, complex symbols have a *structure*. That is, they consist of basic symbols *organized in a certain way*. Even though the complex symbols 'Lassie bit Bloggs' and 'Bloggs bit Lassie' are made up of *exactly the same basic symbols*, they are quite distinct. What makes them distinct is the way the basic symbols are organized. In other words, they are distinct because they have different structures. In Section 6.5 we will examine a powerful argument which aims to establish that thoughts must have a structure. Demonstrating that thoughts are structured is not enough to establish that CTM is true: perhaps thoughts are structured but thinking is not computation. Nevertheless, establishing that thoughts are structured would considerably advance the claim that the mind is a computer.

So far in this section we have garnered a certain amount of support for CTM; however, problems abound. Here I will briefly mention four difficulties which we will later take up in detail.

1. We have seen that computational processes respect the semantic properties of the symbols involved. This raises an important issue: how do the symbols get their semantic properties? The symbols in a conventional computer get their semantic properties from *us*. The 'Fodor' symbols in this document refer to a certain American philosopher *because I say they do*. It is because *I* am thinking about a certain American philosopher that my 'Fodor' symbols refer to the author of *The Modularity of Mind*. I could, if I wanted to, write a document about Russian literature in which my 'Fodor' symbols refer to the Russian novelist Fodor Dostoyevsky. It's my intention to use 'Fodor' to stand for the American philosopher rather than the Russian novelist which determines that my 'Fodor' symbols refer to the author of *The Modularity of Mind* rather than the author of *Brothers Karamazov*.

So far we have seen that the symbols of conventional computers get their semantic properties from the intentions of the humans using the computer. But where do human mental symbols get *their* semantic properties from? There seem to be, broadly speaking, two possibilities. Either mental symbols somehow get their semantic properties from each other, or they get their semantic properties by being related in some special way to things in the world beyond the mind. Neither possibility is without its difficulties. We will return to these issues in Chapter 9.

2. If thinking is computation, then anything which performs the right sort of computations is a thinker. In Section 6.6 we will consider an apparent counterexample to CTM: a set-up in which all the right computations appear to have been performed and yet no (relevant) thinking has gone on.
3. A further difficulty for CTM takes the form of a rival account of mental processes. Called '**connectionism**', the rival account is the topic of the next chapter.
4. Finally, it's not at all clear that CTM has the resources to account for consciousness. Many people have the intuition that a computer could carry out all the right computations and yet not be conscious. I will say a little bit more about computationalism and consciousness in Section 6.6.