Mind, computational theories of

The computational theory of mind (CTM) is the theory that the mind can be understood as a computer or, roughly, as the ‘software program’ of the brain. It is the most influential form of ‘functionalism’, according to which what distinguishes a mind is not what it is made of, nor a person’s behavioural dispositions, but the way in which the brain is organized. CTM underlies some of the most important research in current cognitive science, for example, theories of artificial intelligence, perception, decision making and linguistics.

CTM involves a number of important ideas. (1) Computations can be defined over syntactically specifiable symbols (that is, symbols specified by rules governing their combination) possessing semantic properties (or ‘meaning’). For example, addition can be captured by rules defined over decimal numerals (symbols) that name the numbers. (2) Computations can be analysed into ‘algorithms’, or simple step-by-step procedures, each of which could be carried out by a machine. (3) Computation can be generalized to include not only arithmetic, but deductive logic and other forms of reasoning, including induction, abduction and decision making. (4) Computations capture relatively autonomous levels of ordinary psychological explanation different from neurophysiology and descriptions of behaviour.

1 Syntax, semantics and algorithms

Crucial to understanding computational theories of mind (CTM) is the distinction between the syntax and the semantics of a symbol system. How symbols may be combined comprises the syntax of a system and what the symbols mean or name comprises its semantics. For example, consider three different notational systems for numbers: decimal numerals (‘1’, ‘2’, ‘3’,…), Roman numerals (‘I’, ‘II’, ‘III’,…) and binary numerals (‘1’, ‘10’, ‘11’,…). These different types of numerals name the same numbers (‘5’ names the same number as ‘V’ and ‘101’) but the rules for combining the numerals in calculations are very different. In other words, these systems each have a different syntax, but (approximately) the same semantics.

When children learn arithmetic, they learn reliable rules, or ‘algorithms’, for computing syntactically with numerals (symbols) in ways that mirror their semantics (in particular, the arithmetical relations among the corresponding numbers). Children doing calculations with decimal numerals should get the same answers as a computer working in binary, but the algorithms each uses look very different.

An algorithm is a ‘mechanical’ step-by-step procedure operating on syntactically well-defined symbols in a way that captures relations among the things the symbols represent. Alan Turing developed a general account of algorithms, conceiving what are called ‘Turing machines’, perfectly mechanical computing devices that arrive at a determinate answer or ‘output’ to a question, given certain data or ‘input’ (see Turing machines). According to the influential ‘Church-Turing thesis’ Turing machines can systematically compute anything that is intuitively ‘computable’ (see Church’s thesis). This idea is the inspiration for much of the modern computer industry. CTM extends it to psychology: the suggestion is that all intelligent processes can be systematically decomposed into algorithms consisting of steps that can be executed by primitive processors of a machine.

2 Primitive processors

The specific architecture of a Turing machine involves a 'tape' consisting of an infinite number of individual cells and a 'scanner', whose primitive processes consist in registering whether it is scanning a '1' or a '0', and then moving left and right from cell to cell (see Turing machines). This is only one among many possible computational architectures. What is essential to a computer is merely that the primitive processes be realizable by some physical means or other. To make computational theories of mind (CTM) remotely plausible as an account of the human mind, we will think here in terms of the kinds of electrical devices that seem at least to be available in the human nervous system (but are also exploited in commercial computers).

A good example of a primitive processor is a 'gate'. An 'and'-gate is a device that accepts two inputs and emits a single output. If both inputs are '1's (that is, sum to 2), the output is a '1'; otherwise, the output is a '0'. An 'exclusive-or' (either but not both) gate is a 'difference detector': it emits a '0' if its inputs are the same and it emits a '1' if its inputs are different (or sum exactly to 1).

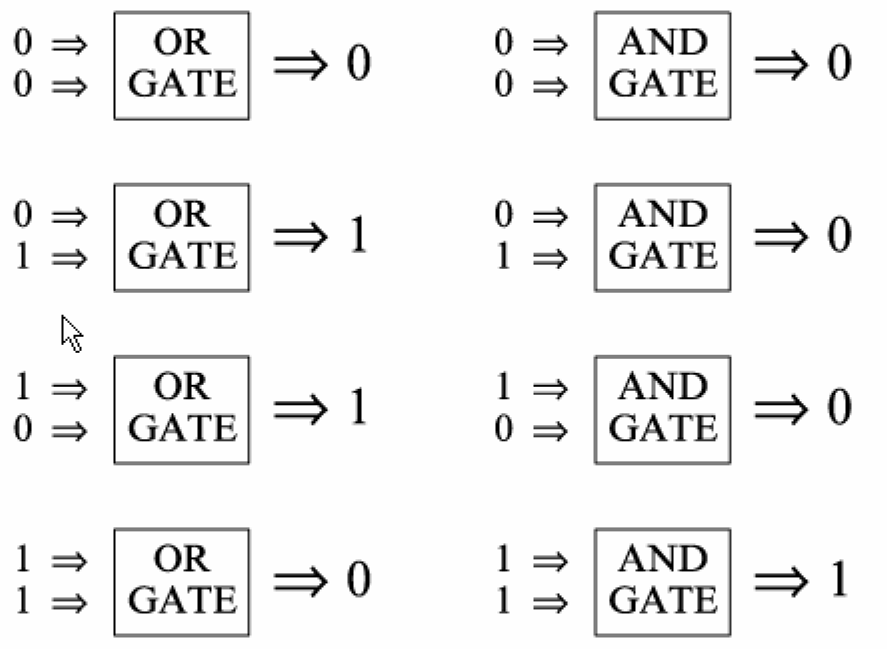

Figure 1a. Firing pattern of an "exclusive-or"-gate *Figure 1b.* Firing pattern of an "and"-gate

This talk of '1' and '0' is a way of thinking about the 'bi-stable' states of computer representers: they are almost always in one or the other of two states; the in-between states are not exploited computationally.

Using these gates, one could construct a binary 'adder' that operated according to the following rules of binary addition:

$$0 + 0 = 0$$
$$1 + 0 = 1$$
$$0 + 1 = 1$$
$$1 + 1 = 10$$

The two digits to be added are connected both to an ‘and’-gate and to an ‘exclusive-or’-gate as illustrated.

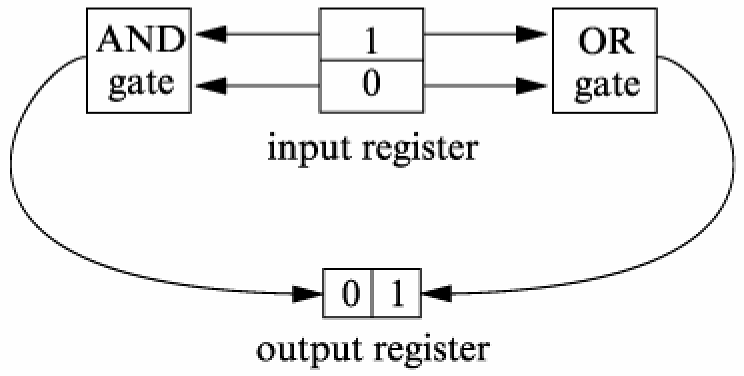

Figure 2a. Adder computing 1 + 0 = 1

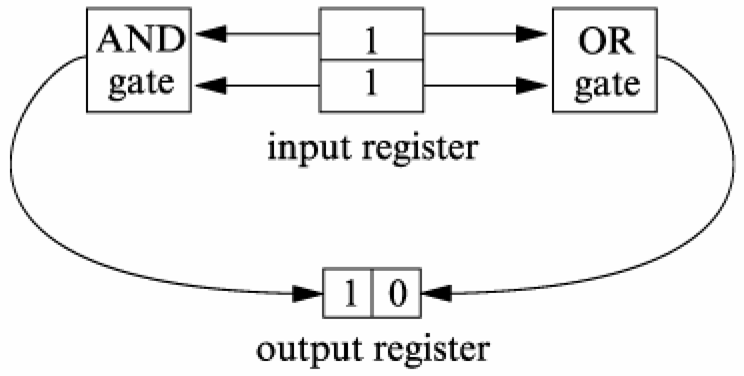

Figure 2b. Adder computing 1 + 1 = 10

Consider figure 2a first. The digits to be added are ‘1’ and ‘0’, and they are placed in the input register (the top pair of boxes). The ‘exclusive-or’-gate sees different things, and so outputs a ‘1’ to the right-most box of the answer register (the bottom pair of boxes). The ‘and’-gate outputs a ‘0’ except when it sees two ‘1’s, so it now outputs a ‘0’. In this way, the circuit computes ‘ ’. For this problem, as for ‘ ’ and ‘ ’, the ‘exclusive-or’-gate does all the real work. The role of the ‘and’-gate in this circuit is ‘carrying’, which is illustrated in figure 2b. The digits to be added, ‘1’ and ‘1’, are again placed in the top register. Now, both inputs to the ‘and’-gate are ‘1’s, so the ‘and’-gate outputs a ‘1’ to the left-most box of the answer (bottom) register. The ‘exclusive-or’-gate puts a ‘0’ in the right-most box, and so we have the correct answer, ‘10’.

3 Syntactic engines

The key idea behind the adder is that of a mirroring, or an isomorphism, between syntactic relations among numerals and their semantics, namely the relations among the numbers the numerals represent (see §1 above). This idea need not be confined to arithmetic. In their development of modern ‘formal’ logic, Frege and Russell showed how deductive logic can be formalized: that is to say, it can be characterized in terms of syntax, and this syntax can be characterized in terms of the physical form of sentences that receive a systematic semantic interpretation. In particular, it can be shown that a certain set of syntactically and formally specifiable rules is adequate to capture a significant class of deductively valid arguments (see Formal languages and systems).

Ordinarily, sentences in logic are manipulated by people consciously following explicit rules, such as modus ponens, that are defined over their syntactic form, and it is often feared that CTM is committed to ‘homunculi’ in the brain executing the rules. However, the rules could also be obeyed by a Turing machine or, alternatively, a machine constructed along the lines of our simple adder: for example, whenever the machine detected sentences of the form ‘P’ and ‘ ’ as input, it could print ‘Q’ as output, thereby obeying modus ponens. In general, Turing showed that there could be a sequence of physical processes that realized any finite piece of deductive reasoning simply as a consequence of its physical organization. Consequently, one’s reasoning could be in accordance with the rules of logic, not by virtue of someone representing and following them, but as a result of the causal organization of the brain. This is what is meant by the expression a ‘syntactic engine driving a semantic one’ (see Fodor 1975, 1980; Newall 1980). As Dennett (1975) put it, intelligent processes are broken down into separate processes so stupid that a mere machine could execute them.

Note that, although the machine’s operations can be defined without mentioning the meaning of symbols, this does not entail that the symbols are meaningless, any more than the fact that bachelors are defined without reference to hair entails that bachelors are bald. Indeed, computations are standardly defined over symbols that are presumed to have some meaning or other (a point often missed by critics of CTM such as Searle (1984: 33); see Chinese room argument).

How do symbols in computers acquire meaning? In the case of most artificial computers, their semantics are pretty much whatever their designers say they are. However, this might not always be so: philosophers have proposed a wide variety of conditions, involving both the internal states of a system, and their (for example, causal and co-variant) relations to external phenomena, that arguably would provide natural conditions for the states of a machine - or of an evolved animal like ourselves - having specific meanings (see Semantics, informational; Semantics, teleological; Semantics, conceptual role). These theories could provide semantic interpretations for the syntactic states postulated by CTM.

4 Other forms of reason

Deductive arguments were the inspiration, and remain the clearest success, for CTM. Similar hopes are entertained by many philosophers and cognitive scientists for a logic of induction, abduction, practical reason and decision making (see Inductive inference; Inference to the best explanation; Rationality, practical). The following affords an extremely simple example of what they have in mind.

Imagine that a computer is provided with programs that perform operations on sentences in a formal language like that of modern symbolic logic, and that it is supplied with certain hypotheses, say, about circles and squares, to which it assigns certain initial probabilities. A video camera presents it with some inputs that directly cause it to access those hypotheses, deducing from each of them, one by one, the consequences regarding the character of the input that would be expected if the hypothesis were true. These consequences are compared in each case with the actual input that is received, and that hypothesis is 'accepted' which satisfies some threshold value of a function of the initial probabilities and the highest degree of fit between its consequences and that input.

Imagine also that the machine is capable of performing certain basic actions: for example, whenever certain sentences (for example, 'Move right!') are in certain 'command' registers, then its 'arms' (say) are caused to move in a corresponding way. And suppose that the sentences that end up in these command addresses are the result of a program that spells out familiar decision-theoretic reasoning ('If more squares are wanted, and the probability of getting some by moving right is greater than by moving left, then move right') that are applied to an arbitrary set of basic preferences that are wired into the machine (say, that it prefers to replicate squares and obliterate circles).

If the machine now operated according to these programs, its actions would seem to be explainable along familiar mentalistic lines. Although reasoning in most animals is massively more complex than this, these programs would seem to capture what is essential to such processes as perception, hypothesis testing, belief, desire and decision making. That, at any rate, is the claim of CTM.

5 Gödel's incompleteness theorem

It is often thought that Gödel's famous 'incompleteness' theorem, which proved that no formal system can prove all the truths of arithmetic (see Gödel's theorems §§3-4), showed that the human mind cannot be a computer, and that CTM must therefore be false (see Lucas 1961; Penrose 1989). After all, Gödel appreciated the truth of the very sentences he is supposed to have proved to be unprovable!

There are a great many problems in this reasoning (for a start, CTM is not committed to all reasoning being deductive), but perhaps the most basic one arises from not appreciating the essentially disjunctive character of Gödel's theorem: what he proved was that either a formal system of arithmetic was inconsistent or it was incomplete. If people managed to do something that counted as proving the relevant undecidable sentence for their own system of arithmetic, they would do this at the expense of no longer being consistent: a contradiction could now be derived from their axioms. But this would not be a particularly remarkable fact: they would not explode or die. They would simply be disposed to believe a contradiction (a disposition many of us possess already). It is true that there are things no machine can do, namely, prove all the truths of arithmetic without contradicting itself; but it has yet to be established that any human being can do them either (see Putnam 1960; Lewis 1969, 1979).

6 Different modes of computation

There are many different ways in which computations can be carried out. Along lines closest to the original computers described here, there are the 'classical', 'symbolic' approaches that typically involve a 'language of thought' that is encoded and manipulated in the hardware of the computer (see Fodor 1975; Newall 1980; Language of thought). But more recently there have emerged 'non-symbolic', 'connectionist' approaches that eschew sentence-like structures, and exploit networks of interconnected cells subject to varying degrees of excitation (see, for example, Churchland and Sejnowski 1992; Connectionism). However, the two approaches need not be exclusive: through ingenious encoding, connectionist programs can be (and standardly are) run on computers constructed along classical lines, and vice versa. This is why one cannot tell simply by looking at the physical architecture of the brain whether the mind 'really' has a classical or connectionist 'cognitive architecture'.

7 Different explanatory levels

This last point highlights an issue central to computational approaches to the mind: that there may be many different explanatory levels for describing the brain (see Explanation). Note, for example, that what is essential to the success of the computations of even our little adder (see §§2-3 above) does not depend upon the gates being realized electronically. An indefinite variety of devices (composed of pipes of water, or of mice in traps) could obey the same rules. Two primitive processors (such as gates) count as computationally equivalent if they have the same input-output function, that is, the same actual and potential behaviour, even if one works hydraulically and the other electrically. So CTM describes the organization of the brain at a level that abstracts from most of its biology.

This is not to say that the computer model is incompatible with a biological approach. Sometimes important information about how a computer works could be gained by examining its circuits. As Lycan (1981) emphasizes, there may well be indefinitely many different levels of description and explanation of a system's processes, within both CTM and biology itself.

CTM is thus a paradigm of a functionalist approach to the mind, whereby mental states are individuated by their role in an organization (see Functionalism). Unlike dualism and physicalistic reductionism, it is not committed to any particular claims about the substance out of which minds might be composed. But, unlike behaviourism, CTM is not entirely indifferent to what goes on inside the head.

In particular, although computational equivalence of primitive processors is defined in terms of their input and output, the equivalence of non-primitive devices is not. Consider two multipliers that work via different programs. Both accept inputs and emit outputs only in decimal notation.

One of them converts inputs to binary, does the computation in binary, and then converts back to decimal. The other does the computation directly in decimal. These are not computationally equivalent multipliers despite their identical input-output functions. Thus, not only might creatures made of different stuff have the same mental lives, but behaviourally indistinguishable creatures might have very different ones.

See also: Artificial intelligence; Chinese room argument; Cognitive architecture; Vision
NED BLOCK
GEORGES REY

Routledge Encyclopedia of Philosophy, Version 1.0, London: Routledge

References and further reading

Bechtel, W. and Abrahamsen, A. (1991) Connectionism and the Mind, Oxford: Blackwell.(The most comprehensive philosophical discussion of the connectionist approach to cognition.)

Block, N. (1990) ‘The Computer Model of the Mind’, in D. Osherson and E. Smith (eds) An Invitation to Cognitive Science, vol. 3, Thinking, Cambridge, MA: MIT Press.(A fuller exposition of CTM.)

Churchland, P.S. and Sejnowski, T.J. (1992) The Computational Brain, Cambridge, MA: MIT Press/Bradford Books. (An excellent introduction to the study of neural computation using artificial neural networks.)

Dennett, D. (1975) ‘Why the Law of Effect Will Not Go Away’, in Brainstorms, Cambridge, MA: MIT Press. (Discussion of how computational processes do not presuppose intelligence to execute them.)

Fodor, J. (1975) The Language of Thought, New York: Crowell.(Influential exposition and defence by a philosopher of the hypothesis that thought consists in computations over syntactically specified, semantically valuable representations encoded in the brain.)

Fodor, J. (1980) ‘Methodological Solipsism Considered as a Research Strategy in Cognitive Psychology’, Behavioral and Brain Sciences 3: 417-24.(Argues that psychology need only concern itself with internal computations over formally specified representations, independent of their external interpretation.)

Lewis, D. (1969) ‘Lucas against Mechanism’, Philosophy 44: 231-3.(A reply to Lucas’ argument against CTM.)

Lewis, D. (1979) ‘Lucas Against Mechanism II’, Canadian Journal of Philosophy 9 (3): 373-5.(A further reply to Lucas.)

Lucas, J. (1961) ‘Minds, Machines and Gödel’, Philosophy 36: 112-27.(One of the original efforts to argue against CTM on the basis of Gödel’s theorem.)

Lycan, W. (1981) ‘Form, Function and Feel’, Journal of Philosophy 78: 24-50.(Useful discussion of how ‘multi-realizability’ recurs at innumerable levels of scientific explanation.)

Newall, A. (1980) ‘Physical Symbol Systems’, Cognitive Science 4 (2): 135-83.(Discussion by a leading computer scientist of the hypothesis that thought consists of computations defined over syntactic objects.)

Penrose, R. (1989) The Emperor’s New Mind: Concerning Computers, Minds, and the Laws of Physics, New York: Oxford University Press.(A recent attack on CTM, based in part on Gödel’s theorem.)

Putnam, H. (1960) ‘Minds and Machines’, in Philosophical Papers, vol. 2, Mind, Language and Reality, Cambridge: Cambridge University Press, 1975.(An early, influential statement of CTM.)

Rey, G. (1997) Contemporary Philosophy of Mind: A Contentiously Classical Approach, Oxford: Blackwell.(An exposition and defence of CTM as an approach to philosophy of mind quite generally.)

Searle, J. (1984) Minds, Brains and Science, Cambridge, MA: Harvard University press. (Discussion of the ‘Chinese room’ objection to CTM.)

Routledge Encyclopedia of Philosophy, Version 1.0, London: Routledge

Artificial intelligence

Artificial intelligence (AI) tries to make computer systems (of various kinds) do what minds can do: interpreting a photograph as depicting a face; offering medical diagnoses; using and translating language; learning to do better next time.

AI has two main aims. One is technological: to build useful tools, which can help humans in activities of various kinds, or perform the activities for them. The other is psychological: to help us understand human (and animal) minds, or even intelligence in general.

Computational psychology uses AI concepts and AI methods in formulating and testing its theories. Mental structures and processes are described in computational terms. Usually, the theories are clarified, and their predictions tested, by running them on a computer program. Whether people perform the equivalent task in the same way is another question, which psychological experiments may help to answer. AI has shown that the human mind is more complex than psychologists had previously assumed, and that introspectively ‘simple’ achievements - many shared with animals – are even more difficult to mimic artificially than are ’higher’ functions such as logic and mathematics.

There are deep theoretical disputes within AI about how best to model intelligence. Classical (symbolic) AI programs consist of formal rules for manipulating formal symbols; these are carried out sequentially, one after the other. Connectionist systems, also called neural networks, perform many simple processes in parallel (simultaneously); most work in a way described not by lists of rules, but by differential equations. Hybrid systems combine aspects of classical and connectionist AI. More recent approaches seek to construct adaptive autonomous agents, whose behaviour is self-directed rather than imposed from outside and which adjust to environmental conditions. Situated robotics builds robots that react directly to environmental cues, instead of following complex internal plans as classical robots do. The programs, neural networks and robots of evolutionary AI are produced not by detailed human design, but by automatic evolution (variation and selection). Artificial life studies the emergence of order and adaptive behaviour in general and is closely related to AI.

Philosophical problems central to AI include the following. Can classical or connectionist AI explain conceptualization and thinking? Can meaning be explained by AI? What sorts of mental representations are there (if any)? Can computers, or non-linguistic animals, have beliefs and desires? Could AI explain consciousness? Might intelligence be better explained by less intellectualistic approaches, based on the model of skills and know-how rather than explicit representation?

1 Historical beginnings

Artificial intelligence (AI) researchers make two assumptions. The first is that intelligent processes can be described by algorithms (sometimes called ‘effective procedures’), which are rules where each step is so clear and simple that it can be done automatically, without intelligence. This is an empirical hypothesis, which some critics of AI accept (Searle, in Boden 1990: ch. 3) and others reject (Penrose 1989). The second is that all algorithms can be implemented on some general-purpose computer. This assumption is generally accepted. It is based on the Church-Turing thesis, which states that a universal Turing machine, to which general-purpose computers are approximations, can compute any algorithmically computable function (see Church’s thesis; Turing machines).

The best-known types of AI - classical AI and connectionism - share these two assumptions. But they differ in other ways (see §§2-4). Classical AI involves serial, or one-by-one, processing of (sometimes complex) formal instructions, whereas connectionism involves parallel, or simultaneous, processing among many simple units. Classical computation uses programs made up of formal rules to generate, compare and alter explicit symbol structures (see Mind, computational theories of; Language of Thought). Connectionist computation typically uses numerical (statistical) rules to determine the activation within networks of locally interacting units and, in systems that can learn, to alter the firing thresholds of individual units and the (excitatory or inhibitory) ‘weights’ on their interconnections. The system’s ‘knowledge’ is contained implicitly in the constellation of connection weights (see Connectionism). Some philosophers uuse ‘computation’ to apply only to the classical type, first defined by Turing in 1936. AI researchers themselves normally use the term to cover both kinds of information processing.

Despite their differences, both these types of AI started from the same source: a seminal article written in 1943 by McCulloch and Pitts (Boden 1990: ch. 1). This discussion of ‘A Logical Calculus of the Ideas Immanent in Nervous Activity’ integrated three powerful ideas of the early twentieth century: propositional logic, the neuron theory of Charles Sherrington, and Turing computability.

The authors showed that simple combinations of (highly idealized) neurons could act as ‘logic gates’. For instance, a McCulloch-Pitts neuron with two inputs could fire if and only if both inputs were firing (an ‘and’-gate), or if only one input were firing (an ‘or’-gate), or if some specific input were not firing (a ‘not’-gate). Since every truth-function can be expressed with ‘not’ and ‘or’ alone, McCulloch and Pitts were able to show that every function of the prepositional calculus is realizable by some neural net; that every net computes a function that is computable by a Turing machine; and that every computable function can be computed by some net. Their work inspired early efforts in both classical and connectionist AI because they appealed to logic and Turing computability, but described the implementation of these notions as a network of abstractly defined ‘neurons’ passing messages to their neighbours.

The neural networks discussed in this entry were extremely simple. For example, any link always had the same amount of influence, whereas most modern connectionist systems allow for continuous changes in the weight of each connection. But the authors’ theoretical ambitions were vast. Perception, reasoning, learning, introspection, motivation, psychopathology and value judgments: all, said McCulloch and Pitts, could in principle be understood in their terms. The whole of psychology would in future consist of the definition of various kinds of nets capable of doing the things minds do - that is, capable of computing the sorts of things which minds compute. Neurophysiology and neuroanatomy would show how networks are implemented in the brain, but psychology would define their logical-computational properties. Their views on the relation between psychology and physiology (or mind and body) anticipated later developments in the philosophy of mind (see Functionalism).

McCulloch and Pitts’ 1943 paper made AI possible in three ways. It influenced von Neumann, in designing the digital computer, to use binary arithmetic and binary logic (see Neumann, J. von). It gave both psychologists and technologists the confidence to model propositional (symbolic) reasoning, as opposed to only arithmetical calculation, on logic-based computers. And it inspired people to start studying the computational properties of various types of neural network. Although classical and connectionist AI are often described as utterly distinct paradigms, research in both these approaches commenced because of this paper.

Early connectionist work was further encouraged by McCulloch and Pitts in a paper of 1947. They pointed out that the brain is a parallel-processing device, not a sequential one. Moreover, it can function acceptably even when some cells misfire or die, or when the input signal is ‘noisy’. The perfect input data assumed within their first paper are, in real life, neither necessary nor often available. The question arises, then, how we (and animals) manage without them. McCulloch and Pitts described a statistical technique, based on differential equations like those of thermodynamics, whereby a parallel-processing system could compute (learn to distinguish) various patterns despite slight variations in the input. These (statistical) ideas were less biologically unrealistic than their earlier (logic-based) discussion. Nevertheless, the 1947 paper was less influential over the next three decades than their earlier work. Only in the 1980s did statistical, parallel-processing models achieve prominence (see §3 below).

2 Classical AI

Classical AI is the best-known type of AI, and is sometimes called traditional AI. It uses sequential programming (do this, then do that), and employs internal representations of lists, semantic networks, arrays and other information-processing structures. These representational structures and their components are interpreted as symbolic representations of propositions and concepts (or beliefs and ideas). Accordingly, this approach is also called symbolic AI.

Most internal representations in classical AI are language-like, being constructed from components each of which has some distinct causal-semantic role (though just which role may vary according to context). Some philosophers, such as Jerry Fodor, explain human mental states, or propositional attitudes, in terms of a hypothetical 'language of thought' having logical properties (compositionality, productivity, systematicity) like those exploited in classical AI (see Fodor, J.A.; Language of thought). A 'toy' example of one simple type of classical AI program (a production system) might look something like this:

If thirsty then set goal to drink.
If current goal is drink and weather is cool then set goal to seek kettle.
If current goal is seek kettle and not in kitchen then go to kitchen and locate kettle.
If kettle is empty then fill kettle with water.
If kettle is full then put kettle on hob and heat hob and locate teapot.
(and so on)

As this toy example suggests, every action, and every condition for action, has to be explicitly specified. Actions that undo previous actions (such as emptying the kettle you just filled) must be avoided. Some unintended consequences of actions have to be anticipated and tidied up (turn off the hob). Default steps must be specified in case any precondition is not satisfied (hot weather, not thirsty). Goal-subgoal structure must be recognized, and the program must be able to 'pop up' to the top goal-level when the lowest sub-goals have been achieved or abandoned. Moreover (what the toy program does not show), procedures must be provided for carrying out the tests (is it cool, and is the kettle full?) and for executing the lowest-level actions (going to the kitchen, locating and filling the kettle) (see Rationality, practical).

Classical AI modelling is widespread in computational research. It is used to study, for example, problem solving, planning, vision, robotics, learning, natural-language understanding, analogy and the perception and performance of music (Boden 1987, 1988, 1990; Rich and Knight 1991). It is applied also to phenomena often assumed to be intractable for a computational (or even scientific) explanation, such as motivation, emotion and creativity.

Among the advantages of classical AI are its ability to represent hierarchical structure and to provide relatively transparent models (whose workings can be well understood by inspecting the program). A further advantage is that it can define 'strong' (exceptionless) problem constraints. It is sometimes forgotten, especially by proponents of connectionism, that strong problem constraints are often needed. For instance, every sentence must have a noun phrase and a verb phrase; and waltz time in music demands that each bar have exactly three beats. Admittedly, a composer may produce some anomalous bars (for example, having only two beats in the upper voice along with three in the lower); but one cannot keep doing this, or break out into march time, without abandoning the goal of composing a waltz. Nor can one communicate intelligibly if one omits most noun phrases. Given that certain rules are mandatory, an AI system should respect them, not approximate them by blurring them with others.

Although it began no earlier than connectionism, classical AI achieved visible success before parallel-processing models did. The first major successes occurred in the 1950s. The logic theorist and general problem-solver of Newell and Simon introduced ‘means-end analysis’, wherein a program analyses the problem as a hierarchy of goals and sub-goals (on indefinitely many levels) and chooses the action most likely to reduce the difference between the current state and the desired state (the goal). This method was widely adopted in theorem proving, problem solving and planning. Another early landmark was Samuels’ draughts (checkers) player, which played well and even learned to adapt to its opponent’s individual style. And early language-using programs used stored English word strings and simple linguistic schemata to conduct ‘conversations’ in which the human interlocutors were occasionally (if briefly) persuaded that they were interacting with another person. (See Feigenbaum and Feldman 1963.)

By the early 1970s there had been considerable advance. For instance, natural-language processing could now be sensitive to highly complex syntactic structure or to the unspoken assumptions hidden in the semantics underlying the actual words - so that programs could ‘answer’ questions about things not explicitly mentioned. Machine learning was sometimes achieved through the program’s having a model not only of the task domain but also of its own action strategies - which, with experience, it modified. Other advances followed. Various high-level AI programming languages were developed, such as LISP (‘list-processing language’) and PROLOG (‘programming in logic’). And Newell and Simon developed ‘production systems’, a programming method based on if-then (condition-action) rules: if the condition is satisfied, then the action is taken. The condition may be a complex conjunction or disjunction, including (sometimes) a statement of the system’s current goal; similarly, the action may be complex and/or internal (see the toy AI program above). These developments affected both technological and psychological AI. Production systems, for instance, are the core of most ’expert systems’, but were originally proposed as a model of human thinking. (An expert system is an AI program consisting of a set of ‘If… then’ rules, which can be used to aid human beings in solving specialist problems such as locating oil, planning a travel itinerary or diagnosing a disease.)

3 Classical AI and human thinking

Traditional AI began with the assumption that symbolic logic is a normative model for both human and automated reasoning (see Common-sense reasoning, theories of). This assumption sits well with some forms of reasoning, such as theorem-proving (although even there, human beings sometimes employ non-logical methods, such as imagery and analogy). But most human reasoning is approximate and qualitative. We can understand speech even when it is ungrammatical, heavily accented and partly obscured by noise; and we can recognize imperfect handwriting, shadowy scenes, and perceptual or linguistic analogies of many kinds. Also, we can use world knowledge in solving logical problems and in deciding whether strictly logical reasoning, as opposed to fallible heuristics based on experience, is appropriate in a given case.

For example, even teachers of logic can solve a problem concerning rules of postage (reversing a minimum number of envelopes to inspect the postage stamps) more easily than a problem of identical logical form posed in abstract terms. In short, we have common sense (see Rationality of belief; Common-sense reasoning, theories of).

Work in AI has increasingly focused on common-sense reasoning, not least because classical AI systems tend to be 'brittle'. If data are missing or corrupted, a classical AI program may give an absurd answer, or none at all. For example, a story-writing program may allow a character to drown a few feet away from a potential rescuer on the river bank, because the programmer did not explicitly include the information that one is normally able to see whatever is going on in front of one's eyes. Similarly, an expert system might ask whether a particular three-year-old girl has any children, not knowing that pre-pubertal girls cannot conceive. People know many such facts about the world and often give something near the right answer (a good guess) even when they lack some relevant information. So AI research has studied probabilistic reasoning, non-monotonic logic (where not-p may turn out to be true even though p had been proved earlier), case-based or analogical explanation, deep (causal) reasoning, and common-sense semantic networks and belief systems (see Non-monotonic logic). As these new methods are incorporated into classical AI programs, the brittleness typical of first-generation AI models (for example, the rule-based 'expert systems' used by many commercial and public institutions) should be reduced.

To reduce brittleness, however, is not to avoid it entirely. Some critics believe that classical AI methods will never model everyday human thinking. A leading AI logicist has recently recanted (McDermott, in Boden 1990: ch. 9). Classical AI has been criticized also by philosophers who reject the Platonist assumption that all truths are analysable in terms of formalizable elements (Dreyfus 1979). Some philosophers see connectionism as immune to philosophical critiques of classical AI (see §3 below), while others regard it as acceptable only up to a point (Dreyfus and Dreyfus, in Boden 1990: ch. 13).

For AI to be deeply relevant to theoretical psychology and the philosophy of mind, it need not promise actual replication of all human behaviour. So someone (such as Fodor) who propounds a computational philosophy of mind may, without contradiction, doubt whether AI systems could ever in practice achieve more than a tiny fraction of human behaviour. Nor need AI researchers themselves believe in this possibility. They may allow that certain behaviour cannot, in principle or in practice, be replicated by AI (of any type). Moods, for instance, may be unreplicable in principle, and superb novel-writing may be unachievable in practice.

Even AI workers who do believe that full replication is possible need not accept the Turing test (see Turing, A.M. §3) as their criterion of intelligence. They all allow that a non-mental thing could sometimes fool us into thinking it intelligent; indeed, this happened so often with ELIZA, an early language-using program, that this user illusion is called the 'ELIZA effect'. And some, for example, argue that mere replication of behaviour, without evolutionary descent of the underlying causal mechanisms, would not suffice for true understanding.

One well-known philosophical critic of AI, especially (though not exclusively) in its classical form, is John Searle (see Boden 1990: ch. 3; Chinese room argument). Searle argues that even if an AI model passed the Turing test (which he thinks is in principle possible), it would not really be thinking, or understanding, anything. Correlatively, a computational psychology could not explain how we can understand: at best, it could explain what we do with meanings once we have them. He argues that programs are defined purely syntactically (by formal rules defined over formal constituents), whereas intentionality - or meaning - involves more than mere syntax. Searle’s argument has been challenged in many ways by philosophers and AI scientists, and remains controversial. Most opponents accept his assumption that computation is purely syntactic, disagreeing with him on other grounds. However, this assumption is also controversial (see Boden 1990: ch. 4). The nature of computation - and its connection with meaning - is less clear, and less universally agreed, than Searle supposes (see Language of thought).

More generally, many philosophical critiques - and defences - of AI turn on issues of semantics (not behavioural replication). But philosophical semantics is itself disputed. To speak of ‘information processing’ is problematic, because the nature of information is controversial (see Semantics, informational). Similar remarks apply to symbol manipulation. Other connected issues include whether meanings are ‘in the head’ (semantic internalism) or partly constituted by the things to which they refer (externalism) (see Content: wide and narrow), and the relevance (if any) of evolution in establishing meaning (see Semantics, teleological). Even within AI, researchers give differing accounts of semantics (and not all AI workers subscribe to ‘strong AI’ as Searle defines it). Some, for example, accept a procedural semantics wherein the execution of particular computations, or mini-programs, suffices for particular meanings (see Semantics, conceptual role). Others see causal interaction with the external world (and even evolutionary history) as necessary for meaning. In short, the semantics of AI are controversial both within and outside the field.

4 Connectionism and hybrid systems

Connectionist models are parallel-processing systems, involving mutually interactive computations grounded in local interactions between connected units (see Connectionism). Each individual computation is much simpler than a typical instruction in a classical AI program. Even so, connectionist units and computations (and learning rules, if any) vary significantly.

For instance, the semantic interpretation given to connectionist units by AI researchers differs. (This is true irrespective of philosophers’ theories about semantics, and the relation of AI and meaning.) Some connectionist units are described as computing the truth-values of entire propositions; others are intended to code familiar concepts. The representational role of these units is thus comparable to the rules and rule components in the toy AI program in §2 (however, connectionist systems cannot specify action hierarchies, like that outlined above for quenching thirst). Yet other connectionist units stand for detailed subsymbolic micro-features, which are not always expressible in terms of everyday concepts or symbols. Thus an input unit might code for a tiny patch of a highly specific (unnamed) shade of purple.

The symbolic/subsymbolic distinction, so defined, is sometimes thought to distinguish (all) connectionist from (all) classical AI. This is a mistake. The distinction is vague: just which concepts count as everyday concepts? Moreover, not all connectionist systems employ subsymbolic processing, and many classical AI programs, especially in vision and natural-language processing, code for subsymbolic microfeatures. The distinction is better defined in terms of the presence or absence of causally efficacious and semantically evaluable logical constituents. Such (symbolic) constituents typify classical AI. Subsymbolic units (in this sense) have no fixed, context-free interpretation, since their effect on processing varies according to the simultaneous activation of the other units.

In the late 1950s and the 1960s, classical AI progressed faster than connectionism. None the less, some researchers persevered with early forms of connectionist modelling. But in 1969 Minsky - widely acknowledged as a father of classical AI, but also the first person to build a connectionist learning machine - proved surprising limitations on what very simple networks (‘perceptrons’) could compute (see Minsky and Papert 1969). Although he stated that more complex networks might be more powerful, there was a marked drop in interest in connectionist research.

A few individuals within AI, and some in psychology and physics, worked on connectionist systems during the 1970s and early 1980s. But not until 1986 did connectionism attract substantial attention. In the late 1980s it hit the headlines and attracted attention from philosophers, being widely hailed as a new, all-powerful computing methodology. More accurately, one specific type of connectionism attracted attention (others being largely ignored): ‘parallel distributed processing’ (PDP). In a distributed connectionist system, a concept is not stored by a single unit (as in localist connectionism). Rather, it is represented by a global pattern of activation spread across the entire network, many different units making some contribution. It may be impossible to assign a specific, context-free interpretation to a given unit. Moreover, no individual unit is either necessary or sufficient for the whole network to represent some particular concept (recognize some particular pattern).

PDP models themselves vary in important ways. Not all employ subsymbolic processing, though most do. Some allow only for on-off variation in the activity of each unit, while others allow for continuous weights defining the unit’s influence. Not all PDP models can learn, though most can. Those which can learn employ various learning rules. The number of units can vary, with significant implications for the type of learning achievable. And although most PDP systems have only ‘feed-forward’ connections, passing from units nearer the input to units nearer the output, some have backward links also (called ‘back-propagation’). In general, PDP systems work by adjusting the simultaneous activity of the constituent units until some global equilibrium is reached (different concepts are represented by different equilibria within the same PDP system). This is achieved not by a sequence of symbolic rules, but by numerically described statistical processes like those of thermodynamics. The changing states of the system are described not as symbol structures (such as lists), but as numerical vectors.

Because of their statistical design, PDP systems are better able than classical models to perform well despite noisy input, and to retrieve an entire memory given only a fragment. Some classical AI programs can do this too, up to a point, if specifically pre-programmed to do so. PDP models, by contrast, ‘naturally’ achieve a plausible end-state given partially conflicting evidence, weighing both strong and weak constraints (and the extent of their mutual coherence). That is, they perform multiple constraint satisfaction, where the information may be partially conflicting and/or missing. And PDP systems with learning rules (which change the activation weights on the connections with experience) can learn –as people can - from being shown a range of examples.

PDP models are unlike real brains in many ways. For instance, the widely used back-propagation algorithm learns in a very un-biological fashion. Less neurophysiologically unrealistic forms of connectionism have been developed, but even these fall far short of neural networks in the brain.

Another drawback of first-generation PDP systems is that, unlike classical AI programs, they cannot model hierarchical structure or sequential processing. Some types of human thinking - many aspects of language and problem-solving, for example - require both these features. Most AI researchers use only classical, or only connectionist, models. This sociological fact has encouraged some philosophers to exaggerate the differences, and the supposed superiority of one approach over the other. However, these AI methods have complementary strengths and weaknesses. Accordingly, there is growing interest in hybrid models, which try to get the best of both worlds.

Various philosophers use connectionist ideas in addressing important philosophical problems. These include symbol grounding, the problem of how words and concepts acquire meaning (Cussins, in Boden 1990: ch. 15); the role of folk psychology in cognitive science (Clark 1989); family likenesses, paradigm cases, and prototypes of concepts (Clark 1993); eliminative materialism (Churchland, in Boden 1990: ch. 14); and scientific explanation (Churchland 1990). Many philosophers see affinities between connectionism and Wittgensteinian views of language, because connectionist representations are not cut and dried like those used in classical AI but allow for borderline cases and for varying degrees of similarity.

5 Situated robotics and anti-representationalism

Classical and connectionist AI share a commitment to internal representation as integral to intelligence. Both these approaches (and most cognitive scientists) posit identifiable data structures ‘in the head’ that are distinguishable from the system’s processing (which is done ‘on’ or ‘with’ them), and that stand for things in the world. This commitment has been (controversially) abandoned by AI work in situated robotics (Boden 1996; Maes 1991). Situated robotics is sometimes termed ‘nouvelle AI’ (in contrast to traditional AI), or ‘behaviour-based AI’ (in contrast to AI based on abstract task decomposition). It claims to be more biologically realistic than classical AI. It emphasizes ‘autonomous’ systems specifically adapted to their environment, not general-purpose computers controllable by many different programs. And it builds whole (sensory-motor) systems, rather than decomposing intelligence into distinct tasks (vision, planning, motor action) which then have to be integrated to provide a functioning robot.

Situated robotics avoids using internal representations of the external (objective) world, although some systems use temporary representations of their own (subjective) place in or actions on their immediate environment. And it uses a bottom-up approach to generate complex behaviour. Because the environment is assumed to be noisy, dynamic and inconvenient, the detailed world-modelling and top-down planning typical of classical AI are rejected. No complex program is involved to decide on, monitor and control the creature’s activities. Instead, the control of behaviour flows from the nature of the system itself, in the sense that the system is engineered (not programmed) to respond to environmental triggers in certain ways. By these means, researchers in situated robotics seek to avoid the notorious ‘frame problem’ (Boden 1990: chaps 7-9). This problem bedevils AI work on robot planning, language understanding and common-sense reasoning. It concerns foresight of the many intended and unintended consequences of action. For instance, if a box is moved across the floor then all its contents move also, whereas the chairs (and table and curtains) do not. A formal representation of action must explicitly allow for all the intended effects, or some may not happen. And the action's many unintended effects will be wholly irrelevant only if the agent is very lucky, or very thorough in explicitly anticipating potentially relevant outcomes.

Classical robots rely on planning, done within an internal world model. To avoid the frame problem, they must explicitly anticipate a host of intended consequences and unintended side-effects. Although this exhaustive listing of consequences is feasible in artificially impoverished environments, it is impractical in the real world (the thirst-quenching program in §2 would often lead to disappointment, because of unexpected facts about the house concerned). Moreover, because the real world cannot be relied on to remain unchanged, detailed anticipatory plans may fail on execution. Instead of manipulating complex internal representations of the world, situated robots deal directly with it. Situated roboticists eschew the abstract functional task-decomposition employed by classical AI. Instead, they analyse intelligence in terms of 'behaviours'. Their robots engage in simple, hardwired behaviour triggered by specific, ecologically relevant, environmental cues.

The anti-representationalist stance of nouvelle AI is controversial. Some situated robots use temporary representations that are not objective but 'deictic' (subject-centred), being closely bound to the robot's behaviour in this place on this occasion (see Content, indexical). Admittedly, some situated robots – including .

References and further reading

Boden, M.A. (1987) Artificial Intelligence and Natural Man, London: MIT Press, 2nd edn, expanded.(A non-technical introduction to AI, including its philosophical, psychological and social implications; extensive bibliography.)

Boden, M.A. (1988) Computer Models of Mind: Computational Approaches in Theoretical Psychology, Cambridge: Cambridge University Press.(A textbook on computational psychology; extensive bibliography.)

Boden, M.A. (ed.) (1990) The Philosophy of Artificial Intelligence, Oxford: Oxford University Press.(Papers on the main philosophical-methodological disputes within AI, with a bibliography.)

Boden, M.A. (ed.) (1996) The Philosophy of Artificial Life, Oxford: Oxford University Press.(Papers on the philosophy of artificial life, with a bibliography.)

Churchland, P.M. (1990) A Neurocomputational Perspective: The Nature of Mind and the Structure of Science, Cambridge, MA: MIT Press.(A defence of Churchland's approach to connectionism, with applications to various philosophical problems including the philosophy of science.)

Clark, A.J. (1989) Microcognition: Philosophy, Cognitive Science, and Parallel Distributed Processing, Cambridge, MA: MIT Press.(An accessible introduction to connectionism with a discussion of its relation to classical AI and of the role of folk psychology in cognitive science.)

Clark, A.J. (1993) Associative Engines: Connectionism, Concepts, and Representational Change, Cambridge, MA: MIT Press.(A more advanced discussion of connectionism and its application to various problems in philosophy and psychology; many references.)

Cliff, D., Harvey, I. and Husbands, P. (1993) 'Explorations in Evolutionary Robotics', Adaptive Behavior 2: 73-110.(A survey of work in evolutionary robotics discussed in §5 above.)

Dreyfus, H.L. (1979) What Computers Can't Do: The Limits of Artificial Intelligence, New York: Harper & Row, 2nd edn.(A philosophical critique of the foundations of classical AI, in relation to Platonic, Cartesian and Continental philosophy.)

Feigenbaum, E.A. and Feldman, J. (eds) (1963) Computers and Thought, New York: McGraw-Hill.(A collection of classic papers in AI, with an extensive bibliography.)

Gelder, T. van (1995) 'What is Cognition, if not Computation?', Journal of Philosophy 91.(A defence of dynamical systems, as opposed to computation, as the basis of mental processing.)

Holland, J.H., Holyoak, K.J., Nisbet, R.E. and Thagard, P.R. (1986) Induction: Processes of Inference, Learning, and Discovery, Cambridge, MA: MIT Press.(Describes how genetic algorithms work, and how they have been applied to various problems, including some of philosophical interest. Fairly difficult.)

Maes, P. (ed.) (1991) Designing Autonomous Agents, Cambridge, MA: MIT Press.(A collection of articles on situated robotics, with many references.)

Minsky, M.L. and Papert, S. (1969) Perceptrons: An Introduction to Computational Geometry, Cambridge, MA: MIT Press.(Limitations on what very simple networks ('perceptrons') can compute.)

Penrose, R. (1989) The Emperor's New Mind, Oxford: Oxford University Press.(An attack on the notion that all human thought can be described by algorithms.)

Port, R. and Gelder, T. van (eds) (1995) Mind as Motion: Dynamics, Behavior and Cognition, Cambridge, MA: MIT Press.(A collection of papers illustrating the dynamic-systems approach in cognitive science, and its critique of representation.)

Rich, E. and Knight, K. (1991) Artificial Intelligence, New York: McGraw-Hill, 2nd edn.(A comprehensive textbook of AI, including detailed descriptions of various AI methods; good bibliography.)

Todd, S. and Latham, W. (1992) Evolutionary Art and Computers, London: Academic Press.(A detailed description of the use of genetic algorithms by a professional artist to produce 'families' of three-dimensional computer sculptures.)

Varela, F.J., Thompson, E. and Rosch, E. (1991) The Embodied Mind: Cognitive Science and Human Experience, Cambridge, MA: MIT Press.(A defence of ’embodiment’ and embeddedness’ in anti-Cartesian cognitive science.)

Routledge Encyclopedia of Philosophy, Version 1.0, London: Routledge

Cognitive architecture

Cognitive architecture involves the properties of mental structures and mental mechanisms that do not vary when people have different goals, beliefs, precepts or other cognitive states. A serious computational theory of mind (CTM) requires that the architecture be constrained independently of such states. One consequence of taking the distinction between architecture and representation-governed process seriously is that it provides a reply to those who are sceptical about the role of rules in cognition, on the grounds that following rules leads to an infinite regress: in CTM, rules are executed by the causal structure of the architecture, and hence do not require further rules for following rules.

The notion of cognitive architecture in the context of the computational theory of mind (CTM) comes directly from the notion of computer architecture, which refers to the relatively fixed set of computational resources available to a programmer in designing a program for a given computer system. Among other properties, this includes the type of memory that the computer has, the way it encodes information (the system of symbolic codes or language it uses), the basic operations that are available, and the constraints on the application of these operations (as in serial v. parallel sequencing). The architecture is a functional characterization of the computer system on which the program runs (see Mind, computational theories of).

It is important to bear in mind that computer architecture reflects the physical properties or ‘hardware’ only indirectly, since the architecture visible to the programmer might itself be simulated in software or firmware. For this reason it is sometimes referred to as the ‘functional architecture’ or even as the ‘architecture of the virtual machine’. Someone writing a program in, say, LISP or C, has available the resources (operations, datastructures and programming constraints) of those languages and is not concerned with their physical instantiations: LISP or C in that caseis the architecture of the relevant virtual machine.

In theories of cognition, the notion of architecture is particularly important because it represents the dividing line between cognitive and non-cognitive aspects of a model. In CTM, mental processes are modelled as computer programs, but the process of executing these programs itself lies outside the domain of CTM. What makes the computer model run, or generate token instances of behaviour, is the causal structure of the underlying computational mechanism, the computational architecture.

The architecture of computational models was often taken for granted in modelling cognitive processes. In the 1950s, information-processing models were typically expressed in terms of whatever architecture was available or seemed intuitively reasonable - usually a so-called ‘von Neumann’ architecture which uses a serial fetch-execute computational regime and location-addressable register memory. But in more recent years, under the influence of Simon and Newell’s literal interpretation of programs-as-theories (see Simon and Newell 1964; developed more fully in Newell 1990), it became clear that the assumptions one made about the underlying architecture strongly influenced which programs it could implement (not which functions it could compute, since with suitable external memory it could compute any computable function) and hence constituted a strong claim about the nature of mind.

Mapping out the cognitive architecture in detail is important if the CTM is to provide a literal scientific hypothesis about the causes of cognitive behaviour. Programs can only be individuated relative to an architecture. According to the way in which computer scientists and cognitive scientists individuate programs, two different architectures cannot execute the same program, although the universality of such machines guarantees that (subject to memory requirements) they can compute the same function. The kind of ‘strong equivalence’ of computational models with mental processes that many cognitive scientists demand is not achieved just by providing a detailed characterization of the process in the form of a program. Because the form of such a program depends on the architecture on which it runs, the theory must specify an independently motivated architecture, including a specification of the representational system it uses. In such a fully articulated model the individual operations - as well as the symbolic expressions they operate over - the sequencing discipline that is imposed on the program execution, the memory constraints that it must adhere to, and so on, all constitute empirical claims. Only when one has independently specified both the architecture and the representations can one’s computational model lay claims to being ‘strongly equivalent’ to the cognitive process being modelled (see Pylyshyn 1984).

In recent years there has been a great deal of debate about which class of architecture is the right one for modeling cognition. In particular there have been proposals that the class of computational architectures that operate over syntactic strings, as in the Turing machine or in proof theory, are inappropriate for modelling psychological processes (see Turing machines; Proof theory). Instead, some people have proposed so-called ‘connectionist’ or ‘neural net’ architectures, consisting of a network of simple threshold elements that compute certain functions (the precise class being unknown) by passing activation among the elements over weighted links (see Connectionism). Such networks, it is claimed, compute without reading or writing symbolic expressions. The debate raises the philosophic issue of whether mind can or should be modelled as a syntactic engine, computing in a ‘language of thought’, as is generally assumed in the CTM (see Fodor and Pylyshin 1988; Language of thought).

Distinguishing architecture and representations (including representations of processes in terms of programs which run on that architecture) is important to the philosopher for a number of reasons (see Pylyshyn 1996). For one, it represents an attempt to understand the nature of the relatively fixed mental structures that instantiate psychological processes: the basic cognitive capacities of the mind within which representation-governed processes are instantiated. Some of these structures are likely to be universal and perhaps even innate (we put aside for now the issue of how architecture changes, except to note that it does not change in response to new knowledge - by definition it is not ‘cognitively penetrable’).

Recognizing the role of architecture also helps to resolve a problem about rule following raised by Wittgenstein (1953): how is a system to know how to follow a rule? If by using another rule, this invites an infinite regress (see Wittgenstein, L. §10). In CTM the regress does not arise because the architecture executes the rules and it does not do this by using rule-following rules, but, instead, by virtue of its causal structure. In a computer we do not have to have algorithms for executing algorithms; instead the physical instantiation of the algorithm in the machine, together with the structure of the machine, simply causes the behaviour to unfold. The relevant abstract description of these causal-structural properties constitutes a description of the architecture of the system.

See also: Computability theory; Language of thought; Modularity of mind
ZENON W. PYLYSHYN