CHAPTER V

ARTIFICIAL INTELLIGENCE: A LIGHT APPROACH

1. GOFAI

Artificial Intelligence is an old dream and a fairly young discipline, which has developed since the late 1950s as an interdisciplinary branch of computer and cognitive sciences aiming at computational models of human cognition. In its strongest version, the one which was predominant for at least the first two decades of its history and is now often labelled GOFAI (Good Old-Fashioned Artificial Intelligence), AI worked on the theoretical basis for the construction of software and hardware, and hence computers and robots, whose behaviour may eventually be at least comparable, if not superior, to the behaviour characterising intelligent human beings placed in similar circumstances, e.g. recognising one's car among many others in the street, driving it to the supermarket safely but at a reasonable speed, parking it successfully, remembering to lock it carefully and inferring that one will not be able to drive it back home if one has left the keys inside when slamming the last door, so one has better call his wife.

To achieve its goal, GOFAI attempted to steer a middle course between

1. *rationalist dualism*, (Cartesianism) according to which
   1.1. intelligence is completely independent from the presence of a biological body—and therefore it would be in principle implementable by other "bio-bodiless" and a-social (stand-alone) forms of cognitive systems, which may not have a biological brain or any interaction with other intelligent beings, for example God, angels, ghosts and computers—but
   1.2. intelligence is also wholly mind-dependent, thus requiring a personal, psychological inner life (the individual "I") so evolved that a vegetable, a machine or even an animal can never achieve it; and
2. *materialist monism* according to which
   2.1. intelligence is nothing else but a complex property of a physical body (epiphenomenalism = mental processes are secondary effects accompanying, and being caused by, physical brain processes in such a way that asymmetric causal relations can link brain states with other brain states and brain states with mental states, but not vice versa)—and therefore it would be in principle implementable by other forms of equally embodied cognitive systems, including animals, extraterrestrial beings and computers—but as such
   2.2. intelligence is also a direct manifestation of life and cannot be disjoined by the whole physical behaviour, bodily experience and natural interaction of a *living organism* both with other living organisms and with its concrete environment within the world, something an engineered machine can never achieve.

The first half of each position is favourable to GOFAI, while each second half makes it unfeasible. As a solution, GOFAI accepted a controversial form of

3. *computational materialism*, not unlike Hobbes', according to which intelligence is biologically body-independent and stand-alone, as in (1.1), but also completely mind-independent, as in (2.1), and therefore it is in principle implementable by a brainless, mindless and lifeless cognitive system enjoying no psychological or bodily experience nor

any interaction with other similar systems.

The only way GOFAI could make (3) a consistent position and support its “computational materialism” was to endorse the following reduction:

4. intelligence = ratiocination = symbolic processing = computation

and adopt, as a criterion of normative assessment, some form of

5. *functional behaviourism*, whose most well-known version is represented by Turing's test.

Programs are formal representations of states and states-transitions. In principle, they are executable by any kind of computational “engine”. Thus, effective computation is indeed mind-independent and can be implemented by any non-biological stand-alone hardware whose performances can then be evaluated by a purely behaviouristic method. The problem is that for over thirty years GOFAI insisted that (4) was not a *reduction* but an *equation*, thus forgetting that intelligence is an utterly individual property, that there are no two intelligent minds identical to each other, that intelligence also evolves as an eminently social phenomenon within a cultural milieu to which we contribute but that is and remains largely independent of ourselves, that a fully developed intelligence requires some powerful semiotic medium which is also socially based, and that not even mathematical thinking can ever be reduced to algorithmic operations, let alone less formalisable processes. It thus committed what Peirce described as a “malignant heresy”:

> [...] first, nothing of importance can be deduced from a single premise, and secondly, from two premises one sole complete conclusion can be drawn. [...] This couple of heresies, being married together, legitimately generates a third more malignant than either; namely, that necessary reasoning takes a course from which it can no more deviate than a good machine can deviate from its proper way of action, and that its future work might conceivably be left to a machine—some Babbage’s analytical engine or some logical machine (of which several have actually been constructed). C. S. Peirce, *Collected Papers*, vol. 4, paragraph 610-611

Research in AI developed as if thinking and behaving intelligently were just synonymous to algorithmic computing, while backgrounds conditions, experiences (feelings, inspirations, passions, intuitions, insights, education, know-how, imagination, sensibility, common sense, taste, aspirations, bodily sensations, consciousness, communication, fears, desires, etc.) and social interactions were not all essential components of an individual's unique intelligent life. Human ability to deal with the world intelligently was seen to be fully and exclusively dependent on human ability to think rationally about the world, and rational thought was seen to be identical with stand-alone, symbolic processing and hence with effective computation. Since GOFAI's regulative ideal was *Star Trek*'s Lt. Comdr. Data, it achieved little but crude caricatures of human intelligence behaviour, and one may wonder what would have happen to the GOFAI research program without the huge military funding—we are talking about hundreds of millions of dollars—that for decades have been made available for strong AI projects, some of which were either aiming at the construction of weapons or directly weapons-related (this is the sense in which some AI projects can raise substantial moral questions in Computer Ethics, in terms of improper, wasteful or even dangerous applications).

2. Turing’s Test

The birth of GOFAI can be dated to the opening of the Dartmouth Summer Research Project on Artificial Intelligence organised by John McCarthy (the developer of LISP, one of AI’s most popular programming languages) in 1956, but it was Alan Turing who had set up the whole agenda for GOFAI several years before (all quotations are from Turing’s “Computing

Machinery and Intelligence”). Turing reduced the question “can machines think?”—which he rightly believed (but too many have since then forgotten) “to be too meaningless to deserve discussion”—to a famous “imitation game”, now known as Turing’s Test:

TT) take a human interrogator IN, a digital computer C and a woman W, place the latter two in separate rooms and make sure they can both communicate with IN only via email (Turing's teleprinter). IN can now ask both C and W all sorts of questions. If IN fails to discover the correct identity of the two interlocutors on the basis of their answers, then C and W are obviously incapable of showing a sufficiently different behaviour.

IN's questions are the input of the system, W's and C's answers its output and the test aims to show the possibility of intelligent machines by applying a weaker version of Leibniz's Law of the identity of indiscernibles: if, *ceteris paribus*, significant differences between W's and C's outputs are indiscernible then W and C are interchangeable, that is, given the same input, the range of outputs a human and a digital processor can generate is such that it makes the differences between the two insufficient for purposes of unmistakable recognition.

Turing Test amounts to an operational criterion to establish the intelligence of a machine. Turing himself thought that

> in about fifty years it will be possible to program computers [...] to make them play the imitation game so well that an average interrogator will not have more than 70% chance of making the right identification after five minutes questioning.

In a superficial way, he was over pessimistic; in a more significant sense, he was definitely wrong. Already around 1964-1967, Joseph Weizenbaum had written and tested a program, called ELIZA, consisting of a language analyser and a set of instructions enabling the machine to “improvise” around a certain topic, mocking, for example, the behaviour of a psychotherapist. A person, let us say IN, could “converse” in English with ELIZA through a typewriter. The reference to Eliza Doolittle, the character in G. B. Shaw’s *Pygmalion*, was not accidental: functions of feedback assured that the program could also “learn” to speak increasingly well. ELIZA was a masterpiece in deception. Here is an example of a typical conversation with a young woman, as reported by Weizenbaum himself (*Computer Power and Human Understanding*):

> YW: Men are all alike
>
> ELIZA: IN WHAT WAY?
>
> They are always bugging us about something or other
>
> CAN YOU THINK OF A SPECIFIC EXAMPLE?
>
> Well, my boyfriend made me come here
>
> YOUR BOYFRIEND MADE YOU COME HERE
>
> He says I’m depressed much of the time
>
> I’M SORRY TO HEAR YOU ARE DEPRESSED
>
> It’s true. I am unhappy
>
> DO YOU THINK COMING HERE WILL HELP YOU NOT TO BE UNHAPPY?
>
> I need some help, that much seems certain

> WHAT WOULD IT MEAN TO YOU IF YOU GOT SOME HELP?
>
> Perhaps I could learn to get along with my mother
>
> TELL ME MORE ABOUT YOUR FAMILY
>
> etc.

ELIZA could carry on similar conversations forever, and indeed, it played this special imitation game so well that people "dialoguing" with the program soon became emotionally involved and tended to anthropomorphise it. After a short time, some practising psychiatrists even thought it might grow into a nearly completely automatic form of psychotherapy. To Weizenbaum's greatest surprise, a large number of people seemed to believe that ELIZA was somehow intelligent. Yet, the software only implemented a limited number of instructions that allowed it to fake a conversation by associating simple questions, standard platitudes or just slightly modified repetitions to particular patterns of input. It was cruising on automatic pilot, as it were, using standard phrases and stereotypes to maintain a state of apparently true interaction—anyone who has dined at a formal meeting or College high table knows very well what this means—but it did not even count as an expert system, for it embodied no knowledge database, and could not be described as being intelligent in any significant sense of the word. Nevertheless, it seems it could pass a decent version of Turing's test. Something had gone wrong.

3. Four Limits of Turing's Test

Although it cannot be the task of this brief section to provide a full assessment of the philosophical and technical implications of Turing's test, it is useful to analyse four of its shortcomings because the history of GOFAI can then be conveniently summarised as the failure to overcome them.

a) *The mimetic ambiguity*.

Turing's test is deceptively simple, but a moment of reflection suffice to make one wonder what it really purports to establish. Does TT show that C's and W's answers can become indistinguishable from each other because the former's resemble the latter's, or rather because the latter's cannot really differ from the former's anymore? The second claim is virtually uncontroversial but also much less interesting than the first: a TT would simply be any irreversible process through which a sufficient loss of information makes the sources of the outputs no more distinguishable or re-identifiable, in the same way as we cannot reconstruct the complex and different inputs of two logic gates from their identical, simple outputs. Turing himself wrote that

> the idea behind digital machines may be explained by saying that these machines are intended to carry out any operations which could be done by a human computer. The human computer is supposed to be following fixed rules [...] we may suppose that these rules are supplied in a book, which is altered whenever he is put on to a new job. He has also an unlimited supply of paper on which he does his calculations.

Ideally, a meticulous and patient scribe, a kind of tireless idiot savant, provided with an unlimited amount of time and stationery, could do everything any Turing machine or classical computer can do. That in most cases he would arrive too late or not understand what he is doing—as this is the case with Searle's Chinese room example, in which the scribe successfully operates on Chinese symbols he does not understand—is explicitly not the point addressed by

TT. Nobody can justifiably disagree about the possibility of a computing scribe, but then, isn't this enough to explain why IN is unable to establish whether C and W are (i) two computers, (ii) two persons acting as two computers, or (iii) a mixed couple, one of which is a computer and the other one has no option but to act as if he were one? From the way in which Turing sets up his test:

- only written communication
- possibly only yes/no questions ("We are of course supposing for the present that the questions are of the kind to which an answer 'Yes' or 'No' is appropriate")
- no critical questions requiring C's or W's personal approach, such as "What do you think of Picasso?" (this is Turing's example)
- no context-sensitive questions and hence irrelevance of the environment
- no symmetrically interactive dialogue, since W and C cannot ask questions but only answer them, cannot exchange their roles nor can C and W be asked to pretend to be other "people" (it may be argued that a computer can mimic some human behaviour, but a human being can mimic any behaviour, no matter how badly)
- no linguistic-pragmatic context (can a computer master the rules of Grice's conversational implicature, whereby a sentence that literally means p is actually used to communicate a message other than p?)

the lack of significant differences between W's and C's answers may well be caused by the downgrading of the human agent to the level of the computer, rather than vice versa. TT would be a "blurring machine", aiming at showing the possibility of W's human stupidity rather than C's artificial intelligence, especially if we also endorse (b).

b) *The reductionist assumption*.

TT-conditions are so constraining that all intelligent behaviour of both C and W is strictly reduced to symbolic processing tasks. No pattern or sound recognition skills are taken into account, for example, since C would not be able to appreciate the difference between *Guernica* and *The School of Athens*. Is this restriction truly irrelevant when the general conclusion that

> We may hope that machines will eventually compete with men in all purely intellectual fields.

is formulated on its grounds? Prima facie, it would seem obvious that "purely intellectual fields" encompass much more than just the capacity to handle well-formed strings of symbols according to finite sequences of instructions. Indeed, one may even argue that the latter capacity is not central to many intellectual skills. In either case, the test is not sufficiently cogent to support the much more substantial conclusion drawn by Turing. Things get even worse once we bring to light (c).

a) *The linguistic fallacy*.

TT presupposes that IN, W and C all speak and understand the same language, otherwise they would not be capable to play a convincing part in the test. This is clear if one recalls that, to introduce TT, Turing presents first another version of the test, in which the actors are a man and a woman, and IN is shown to be unable to discover the sex of his interlocutors. Of course, the language in question must be so powerful, rich and hence "natural" as to allow a satisfactory level of conversation between IN and W and provide no clues about the artificial nature of C. Let us call such a language TT-adequate. To be TT-adequate, the linguistic-semiotic skills of C already presuppose an intelligent agent, whose behaviour is superior not only to that of a brainless coelenterate such as a jellyfish, but also of mammals such as dolphins and chimpanzees, which are known to be able to learn only an extremely limited range of linguistic skills, insufficient to pass the test. There follows that the very mastering of such skills by C already counts as a significant instance of a fully-grown human intelligence. After all, it is since Plato that the process of thinking is described dialectically as inner discourse rather than mathematically. It was only Hobbes who later suggested a comparison with computing, under the influence of the Cartesian revolution:

> For seeing life is but a motion of limbs, the beginning whereof is in some principal part within, why may we not say that all automata (engines that move themselves by springs and wheels as doth a watch) have an artificial life? For what is the heart, but a spring; and the nerves, but so many strings; and the joints, but so many wheels, giving motion to the whole body, such as was intended by the Artificer? [*Leviathan*, Introduction] […] WHEN man reasoneth, he does nothing else but conceive a sum total, from addition of parcels; or conceive a remainder, from subtraction of one sum from another: which, if it be done by words, is conceiving of the consequence of the names of all the parts, to the name of the whole; or from the names of the whole and one part, to the name of the other part. And though in some things, as in numbers, besides adding and subtracting, men name other operations, as multiplying and dividing; yet they are the same: for multiplication is but adding together of things equal; and division, but subtracting of one thing, as often as we can. These operations are not incident to numbers only, but to all manner of things that can be added together, and taken one out of another. […]In sum, in what matter soever there is place for addition and subtraction, there also is place for reason; and where these have no place, there reason has nothing at all to do. […] For reason, in this sense, is nothing but reckoning (that is, adding and subtracting) of the consequences of general names agreed upon for the marking and signifying of our thoughts; I say marking them, when we reckon by ourselves; and signifying, when we demonstrate or approve our reckonings to other men. [*Leviathan*, Chapter 5]

Consider now the following example. One day, some fascists went to see Picasso. Referring to *Guernica,* they asked him: "did you do it?", he replied "No, you did". Apparently, the reply was good enough to save Picasso any further trouble. Would a computer be able to understand a similar dialogue? It would have to, if its language is to be TT-adequate, but if C is granted such a degree of linguistic competence surely the test loses all its probatory value, as it is simply begging the question: technically speaking, full linguistic competence is recognised to be an AI-complete problem, i.e. a problem whose solution *presupposes* a solution to the "strong AI problem", namely the synthesis of a human-level intelligence. Therefore, it would be equally valid to argue that, since C cannot have a behaviour as intelligent as W's it cannot be granted fluency in any TT-adequate language. In each case, a processor capable to master a natural language well enough to play TT will also show a sufficiently intelligent behaviour. The problem then becomes whether we can justifiably assume that C is already fluent in a TT-adequate language. The answer must be in the negative, of course, unless one wishes to maintain only that it is *logically possible* that C may come to master a TT-adequate language. In which case, the whole test comes to be based on (d).

a) *The modal indeterminacy.*

Turing's test shows that "carbon chauvinism" is unjustified: thinking machines are *logically possible*—there is nothing contradictory in the hypothesis that a brain may not have a neuro-physiological nature but an electro-mechanical one, or that a machine may be able to speak a natural language properly—hence GOFAI cannot be ruled out as being impossible in principle, yet this is very far from establishing that silicon intelligence is also even vaguely plausible, likely to evolve, or *empirically feasible*. The distinction is crucial, but it is often overseen. It is one thing to prove that doing p involves no contradiction (hence it is logically possible), take for example the possibility of guessing the whole list of all the single figures indicating how many people used the London underground per day since it was first open, and an utterly different thing to infer that doing p is therefore possible, without further qualifications. Miracles, such as living forever, though extremely improbable can still count as logically possible, it is just that nobody in these days is inclined to develop scientific theories on their likelihood. Now, Turing presented TT only as the basis for a conjecture. He honestly recognised he had no positive arguments in favour of the actual possibility of GOFAI, only negative ones against its critics. Consequently, he was able to show that it was a mistake to consider non-biological intelligence logically impossible. In computer science, however, indirect proofs of purely logical possibilities bear no weight, for only constructive methods are truly effective and after so many failures there seems to be some clear evidence that true AI may not be empirically feasible for all practical purposes. The reply may be that GOFAI projects just need more time to develop. So here is an argument against GOFAI empirical feasibility that relies on the complexity of the algorithms required by any real AI application and the standards of reliability that we have come to expect in all other areas of engineering. Given the incidence of faulty software (ironically, we are used to pay for updated versions, which are not necessarily better, often only because we are sold software whose bugs had still to be discovered; this is so true that software is one of the few common goods that is not usually provided with a warranty, thus showing all its potential unreliability) and how increasingly prone to failure are computers the more sophisticated they become (how many times did your computer crashed in the last six months?), it is unlikely that AI system will ever become sufficiently reliable. Testing and debugging—attempting to detect, specify and eliminate the possible causes of programs' malfunctions— is not a science, is a Sisyphist art, for a bug-free software is almost a contradiction in terms. The more complex a software has to become, to cope better with a larger variety of evolving tasks, the higher is the occurrence of serious programming errors (a serious bug does not generate some misbehaviour, is the end of any behaviour, like a stroke). Unfortunately, the scale of combinational complexity of any non-trivial software (programs with more than 100,000 lines of code; consider that automatic teller are often run by software containing more than 700,000 lines of code, and common software for Windows can easily contain more than 500,000 lines of codes) is so high, that errors cannot be exhaustively detected (let alone fully fixed) by means of empirical tests, for the latter require time-resource, and the computational complexity of the software in question is so great—a complex software can exists literally in billions of discrete states, each state could be the source of an error and hence a *catastrophical* failure in the Greek sense of the word, i.e. a sudden collapse—that we soon run out of it (the process requires amounts of time longer than the life of the universe to be feasible). Of course, a completely foolproof debugging universal program D would be able to determine whether any application A terminates successfully, for all initial situations, but trying to prove that D is possible would be equivalent to trying to prove that there is a program that solves the halting problem, and we have seen in chapter two that this is impossible. The empirical fallibility of software is here to stay. Errors detection is a matter of dedicated testing programs, which are not in themselves error-proof and may or may not discover all the main faults of the software in question. Once some bugs are discovered, and this often happens only thanks to the extended use by millions of users, still because of the complexity of the original software, the probabilities of introducing new severe errors when debugging it are known to be so high, that it has become common practice to correct only the smallest possible fraction of the original errors, and then, rather than tampering further with the software, warn users of any other discovered risky circumstances in which the software may not work properly, in the same way as every drug indicates its potential contraindications, i.e. the potential negative results that may be caused by the complexity of its interactions with the body of the patient and other secondary effects. Now a GOFAI application requires software so complex (billions of lines of code) that it would be "undebugable" in principle, and therefore never absolutely reliable in practice. Mad computers behaving erratically could become a reality. To have some idea of the scale of the problem consider that the computerised system of the space shuttle contains ca. 25,500,000 lines of code, whose development required some 22,100 man-years, but that the whole computer system it is not even vaguely close to what would be a fully artificial brain such as *2001*'s Hal. Currently, there are no existing techniques that deliver software of fully warranted quality and reliability. The solution may lie in the development of error-proof programming languages and programming techniques, and much more powerful tools for software verification and derivation (derivation is verification on-the-fly, a method of showing that the software works properly while is being written), but much as both areas may bring fruitful results in the future, verification and derivation programs can currently handle software of only very modest size, and none of the two approaches is likely to eliminate the substantial problem. For verification, if applied consistently, would limit to much the freedom of the programmer to the point of becoming a serious hindrance to any high-quality GOFAI application, while derivation can only succeed in limiting the damages but never eliminate the chance of some catastrophic dormant bug.

4. The Application-areas of AI

GOFAI developed as the project of empirical research that aimed to make Turing's claims come true. It was based on questionable assumptions—the logical possibility of non-biological intelligence—and not very realistic expectations, so it could not succeed in his most ambitious plans, despite the huge amount of human and financial resources to which it helped itself. This does not mean that AI has not achieved remarkable results. In the course of its attempts to construct a TT-adequate computer that could perform tasks usually supposed to require some form of human intelligence, research in artificial intelligence has come to be organised around the following main areas:

1. Theorem proving.

Historically, this is the first and most successful application of AI and yet, in so far as decision procedures are known for the theorems in question, as in the case of propositional logic theorems, most scientists and philosophers would not consider it real AI, since such theorems could also be proved by a well-constructed windmill or our dull ("brainless") scribe. On the other hand, theorems for which no mechanical procedures are known result much more intractable, and non-computable solutions are based on ad hoc algorithms designed by human experts and intelligent programmers. So much so that when the latters are no longer available, as it is often the case after some time, if there is no documentation and the software needs to be updated it is easier to start again from scratch then try to interpret what the experts had in mind and actually did.

2. Game playing.

In this case too, games for which a decision procedure is known, such as noughts and crosses (tic-tac-toe) or draughts (checkers) are easily implementable, but will hardly be considered instances of full artificial intelligence. Other board games, such as chess or monopoly, require databases and the discovery and application of heuristic procedures, i.e. rules of thumb and general problem-solving techniques. The issue then becomes an empirical one and needs to be further analysed, as we shall see in a moment.

3. Information processing.

This is perfectly achievable in terms of processing of strings of symbols, such as in word-processing, mathematical programs, parsing software or statistical applications, as we saw in chapter four. When understanding is in question then see (6).

4. Planning for robot behaviour via means-ends analysis.

Means-ends analysis is a computing technique in AI to achieve goals by formulating plans. The system implements a plan, constituted by a sequence of actions, and the sequence is constructed by comparing the goals that each action achieves (means) with the goals and action predictions that must be achieved (ends). This is perfectly achievable as long as the environment and the tasks are limited and can be fully formalised. We shall discuss this point more in full later in the chapter.

5. Expert analysis and advice in applied fields, such as medical diagnosis.

This is feasible as long as the expectations are moderate. For example, in medicine, expert systems have been developed that analyse the disease symptoms, medical history, and laboratory test results of a patient, and then suggest a variety of possible diagnosis to the doctor.

6. Fluency in a natural language (building a system that can understand and speak a written and spoken natural language as well as a human being).

Some restricted sectors of a natural language or specific formal languages, languages with identified meaning in fixed contexts, can be understood and processed by machines. There are programs that enable a computer to generate grammatically correct sentences, to establish links between data inputs/outputs and concepts, to recognise written or spoken information and to produce summaries, answer specific questions, or redistribute information to users interested in specific areas of this information. However, interpretation and fluency in a natural language is a human activity par excellence that our present computers will never achieve. The use of a natural language is a constantly evolving ability, which is made possible by, and is intrinsically rooted in, our complex experience of a physical and psychological life and on our being human agents interacting with the world from inside a human body, within a specific human society and a determined culture. Above all, being able to use a language means being able to shape the world of experience. TT takes the possibility of a computer's linguistic proficiency for granted, but while phonetic and graphical features (the physical aspect) and lexical and syntactic aspects (the logical structure) of natural languages have been subjected to formalisation rather successfully and there are hopes that they may be fully controlled by a computerised system, semantics and hermeneutics, i.e. the problems of meaning, understanding and interpretation, have remained largely intractable and appear to be beyond any translation on terms of functins effectively computable by a Turing Machine. Here it is important to recall that a promising program of research in applied computational linguistics, namely statistical semantics (roughly speaking, the computer suggests the most reasonable meaning of a word W by analysing the most consistent system of N words forming a cluster around W), shows the limits of GOFAI (in this case the full simulation of human language competence) and the value of a more pragmatic approach to AI in the area of automatic translation of unrestricted texts and in the area of computer understanding and generation of natural language and hence human-computer communication. Computers cannot master a language like a human being, but software systems can greatly simplify the work of human translators and improve our present interfaces. One may easily think of a number of successful applications such as voice recognition programs, automatic grammar checkers for word processing software, intelligent email filters and routers, text classification systems, systems for information extraction from semi-standardised texts and so forth.

7. Informal, non-deductive reasoning and universal problem solving.

This would require computers to be able to argue or reason, to hold and defend views. It is a largely intractable problem without resort to human intelligence and ad hoc programming.

8. Visual and acoustic perception and pattern recognition.

Pattern recognition systems are currently able to classify data (patterns) thanks to either statistical information, extracted from the patterns themselves, or a priori labelling information provided as previous input. They consist of three basic components:

- a sensor, which collects the data to be described/classified
- a feature-extraction mechanism, which computes symbolic or numeric information from the collected data, usually using one or more of the following approaches: statistical (or decision theoretic), syntactic (or structural), or neural. The statistical approach is based on statistical characterisations of patterns, assuming that the patterns are generated by a probabilistic system. The syntactic approach is based on the structural interrelationships of features. Neural pattern recognition employs the neural computing paradigm that has emerged with neural networks (we shall discuss pattern recognition more in details in section 9.2)
- a classification or description scheme, which describes or classifies the pattern, relying on the extracted features. The classification or description scheme is usually based on the availability of a set of patterns that have already been classified/described (groups of measurements or observations, defining points in an appropriate multidimensional space). A learning strategy based on training sets of a priori labelling patterns is characterised as supervised, but learning can also be unsupervised when the system itself extracts the patterns from the statistical regularities of the observed data.

Perceptual tasks (the acquisition, classification or description of observations) are, in different ways, performed by all forms of life, but have proved to be very difficult for digital machines, except in fully formalisable instances. In particular, computer vision (building a system that can see as well as a human) remains a recalcitrant problem, despite its military importance (note that for a missile "pattern recognition" really means "target individuation").

9. Experience, common sense, self-consciousness, introspection, awareness of and detachment from time-space situations, knowledge, understanding, sense of pertinence, insight, judgement, evaluation, imagination, feelings, taste, intentionality, interpretation.

These are all largely intractable phenomena, since it is still unclear whether and how knowledge—as opposed to data— and other cognitive or mental states may be analysed in such a way that a computer could be said to "understand" or "feel" something and hence apply its "understanding" or "insight" to relevant cases with at least some good common sense. We shall see that this is an extreme case of the kind of problems addressed by knowledge engineers when designing knowledge representation and automatisation systems, and devising general rules and algorithms for the automatic recognition of meaningful and relevant information, in connection with large database.

10. Self-reproducing and self-learning.

Both seemingly intractable. Intelligence appears to have its own entropy: below a certain threshold, a system is able to construct only a range of more simplified systems, whereas a brilliant child can have very dumb parents. As for self-learning, when understood as a general capacity, comparable to that of a child, AI systems as we know them have proved to be incapable of achieving it. A Universal Turing Machine is an "innate" and "orthodox" machine: unable to learn by dialoguing with the environment, it can nevertheless deal with situations that sufficiently approximate its internal microworlds, as long as this does not require to behave heredoxly, i.e. in a way that is neither random nor in accordance with pre-established patterns of rules or instructions, but adequate to the novelty of the situation.

5. The Conditions of Possibility of AI and the Paradox of GOFAI

Based on the previous analysis, the crucial conditions that make AI projects more or less successful can be conveniently summarised under five headings:

a) **EFFECTIVE COMPUTABILITY** (see chapter two).
b) **EPISTEMIC INDEPENDENCE**, i.e. whether either no knowledge/understanding is relevant, or all relevant knowledge/understanding that is presupposed and required by the successful performance of the intelligent task, can be discovered, circumscribed, analysed and made fully manageable through computable processes.
c) **EXPERIENCE-INDEPENDENCE**, i.e. whether either the task is based on universal and "timeless" instructions carried out by the system, or all practical experience, both relevant as a background condition and necessary for the successful performance of the intelligent task, can be discovered, circumscribed, analysed and made fully manageable through computable processes.
d) **BODY-INDEPENDENCE,** i.e. whether either the intelligent task can be performed by a disembodied, stand-alone intelligence, or all "perceptual intelligence", both relevant as a background condition and necessary for the successful performance of the intelligent task can be discovered, circumscribed, analysed and made fully manageable through computable processes.
e) **CONTEXT-FREEDOM**, i.e. whether either the context is irrelevant, or all relevant information concerning the context within which an intelligent task is performed, and which indeed make the task intelligent, can be discovered, circumscribed, analysed and made fully manageable through computable processes.

Points b/e state that a classical computer can cope with intelligent tasks, which necessarily require knowledge, experience, bodily involvement and social interaction with the context when performed by a human being, only by internalising all relevant data, i.e. by transforming the strongly constrained domain of application into a microworld subject to Leibniz's law of pre-established harmony. The specific construction of a microworld "within" a computerised system represents a combination of ontological commitments that programmers are both implicitly ready to assume when designing the system and willing to allow the system to adopt. A computer is always trapped within a microworld. The broader its ontological environment, the less obvious is the spectrum of ontological preclusions engendered by the system, the easier it is for the user to be caught within its limits of scope and domain, the more difficult it becomes to realise the inflexibility of the machine's real ontological commitments and apparent interactions. There are many cases, however, as when we try to implement common sense notions, linguistic skills or learning abilities, in which intelligence proves to be an ontologically diatopic property, that is a general way of behaving and thinking flexibly, constructively and in dynamical interconnection with the surrounding reality, which is not bounded to a specific and constrained area of application. In these cases, to satisfy conditions b/e, the internalisation would require the computer to be nothing less than potentially omniscient, i.e. to have a database containing at least all the rules for generating, and taking into account, all possible microworlds. The paradox behind GOFAI appears then obvious: the better a task fulfils conditions (a)/(e)—as this is often the case in domains of application which are the result of constructions out of explicit rules (board games, well known fields of mathematics, specific databases, etc.)—the more easily it can be "stupefied" and hence the more feasible the relevant

AI project becomes, but then the less we also are inclined to connote as *necessarily intelligent* the agent (not the task) capable to perform it. A good example is provided by a machine capable to demonstrate theorems in propositional logic. On the other hand, the less a task fulfils such conditions—the more sensitive and "open" a task is to completely occasional novelties, requiring non-formalised, non-procedural creative thinking, grasping of the appropriate context, mastering of the relevant information, possession of the right experience, enjoyment of a bodily involvement, social interactions, inventiveness, sensibility, intuition, understanding, linguistic skills and so on—the more the agent capable to perform it is expected to be truly intelligent, but then the less feasible the relevant AI project becomes. Clearly, as the nature of intelligent tasks ranges from the total "stupefiability" of games such as noughts and crosses or draughts, to the complete openness of abilities such as mastering a natural language and laughing to a good joke, the development of successful AI projects becomes a matter of degrees and of empirical trial and error. What a computer can do empirically, not just in principle, is a practical question, whose answer depends on the nature of the specific task under examination and on whether there may be efficient methods to analyse it in such a way as to make factors (a)/(e) sufficiently liable to an AI treatment. GOFAI committed the mistake of considering, a priori, the whole spectrum of intelligent tasks as its own domain, because it started from the wrong end of it, namely Turing's assumption that C is already capable to manage a TT-adequate language. Contrary to Strong AI, Light-AI (LAI) places itself more pragmatically at the other, "stupid" extreme of the spectrum, and tries to advance as far as it is empirically feasible towards its "open" end. The perspective is radically different and so it is the approach to what an "intelligent system" could be like.

6. From GOFAI to LAI

GOFAI endorsed Turing's suggestion that, since a single machine should be able to cover the whole spectrum of intelligent tasks, the best thing for it was to simulate the only other "engine" capable of a similar undertaking, namely the brain/mind. However, GOFAI was not only mimetic, it also set things upside-down. To see why it is useful to draw an analogy with the history of flying machines. It was wrong, if natural, to think that, could humanity one day fly, it would have done so somehow like Ikarus, using a sort of flapping wings. As long as aeronautics tried to construct bird-like vehicles, however, it did not score a single success. To become truly achievable, the task, flying from one place to another, had to be detached from the particular manner in which birds performed it and pursued in ways more amenable to engineering solutions, thus adopting a *functionalist* approach:

- identification of a set F of relevant functions performed by the system S_1
- abstraction of F from the specific nature of the system S_1
- identification of the necessary and sufficient conditions C for the implementation of F
- implementation of C by means of a new system S_2, capable to perform the relevant functions listed in F.

Since GOFAI could start from an ideal prototype, i.e. a Universal Turing Machine, the mimetic approach was also sustained by a re-interpretation of what human intelligence could be. Thus, in Turing's paper we read not only that

i) digital computers must simulate human agents,

but also that

ii) they can do so because the latter are, after all, only complex processors (in Turing's sense of being UTMs).

The suggestion was that all (at least seemingly) cognitive entities (AI applications, people and other living organisms) are computational systems in the "effective" sense of computation seen discussed in chapter two, which carry out their operations by manipulating symbols and transforming, through sequences of internal steps, static input representations (patterns of activity) into other static output representations. It is as if, having constructed a mongolfier while trying to enable man to fly like a bird, one had decided to fill all birds with hot air or hydrogen to make sure any flying system behaved uniformly like a balloon. A neat solution, yet completely wrong. As I have argued in chapter two, even if the brain is a "computing machine" this does not mean that it is necessarily comparable to a classical computer (i.e. a Universal Turing Machine), so (ii) above is at least controversial. Although we need to remember that GOFAI developed in a context in which crude IQ tests were very popular, it is still surprising to see how an empirically minded culture could then be taken so astray by its materialist project of a thinking machine to forget that the physical nature of the brain does not prove that it functions, technically speaking, like a binary data-processor with a program following sets of pre-determined instructions establishing how strings of symbols need to be processed sequentially by logic gates, that there are no ultimate reasons to believe that intelligence is a brain rather than a mental feature and that "mind" is just another word for "brain", and finally, that human knowledge and understanding do not resemble information processing phenomena very much. Attempts to shift the paradigm in favour of parallel, interactive, complex, digital, analog, networked or quantum systems show how unsatisfactory the GOFAI picture is. The generic possibility of modelling the brain as an input-process-output device, often useful for explanatory purposes, was confused with its actual nature and the failure of GOFAI was then the failure of a rather primitive, epistemological model of human intelligence.

7. The Cartesian Nature of LAI

LAI has slowly risen as a phoenix from the ashes of GOFAI. While the latter attempted to "stupefy" the mind assuming an elementary computational model of the brain itself, and hence was bound to fail, the former aims to re-model the specific tasks in question according to the logic and functions of the machine ("stupefication" of the process), so it has been far more successful. The point is that LAI is *constructionist*, not *mimetic*. It recognises that the same tasks can be accomplished equally well in completely different ways and that the difficulty lies in discovering whether, given a specific task, there is a "computer-way" to accomplish it satisfactorily. Therefore, it attempts to engineer problem-oriented applications that do not necessarily simulate but rather emulate (do often better, although differently) what a human being could achieve in the same situation. We do not want a program that can mechanically prove a logic theorem exactly as a human being, because we do not need either the latter's mistakes and distractions, or his or her insights and intuitions. The problem of images recognition provides a useful example. Infrared radiations are electromagnetic radiations of wavelength between about 0.75 micrometers and 1 millimetre. All bodies above the absolute zero of temperature absorb and radiate infrared radiations, which are invisible to the human eye. Above a certain threshold, we perceive infrared radiation as heat, but a computer can easily process infrared wavelengths well below such threshold and show them on a screen as just additional colours. Thus, infrared radiation is used in many fields such as medical photography and treatment, chemical analysis, military applications and astronomy. It is a "smart", LAI computing solution that solves a problem GOFAI found intractable.

Both GOFAI and LAI endorse a Cartesian-like problem-solving procedure but only the latter is correctly definable as Cartesian in spirit. Like Descartes, but unlike GOFAI, LAI rejects the feasibility of a thinking machine capable to clone human intelligence. Nothing could be more illuminating and contrary to Turing's project than the following statement:

> We can certainly conceive of a machine so constructed that it utters words, and even utters words which correspond to bodily actions causing a change in its organs (e.g. if you touch it in one spot it asks what you want of it, if you touch it in another it cries out that you are hurting it, and so on). But it is not conceivable that such a machine should produce different arrangements of words so as to give an appropriately meaningful answer to whatever is said in its presence, as the dullest man can do. Secondly, even though such machines might do some things as well as we do them, or perhaps even better, they would inevitably fail in others, which would reveal that they were acting not through understanding but only from the disposition of their organs. For whereas reason is a universal instrument which can be used in all kinds of situations, these organs need some particular disposition for each particular action; hence it is for all practical purposes impossible for a machine to have enough different organs to make it act in all the contingencies of life in the way in which our reason makes us act." Descartes, *Discourse on the Method*.

The possibility of increasing the number of "stupefiable" tasks, together with the radically constructionist (non-mimetic), non-materialist and Cartesian nature of LAI, explain some of the specific successes and orientations of LAI and clarifies an important mistake often made in the anti-AI literature. We shall first concentrate on the latter to introduce the former.

8. Deep Blue: a Cartesian Computer

Computers cannot play chess better than a Great Master: even a critic of GOFAI as acute as Dreyfus has made the mistake to argue for such a view. Today it is easy to point out that, by defeating Kasparov, IBM's Deep Blue has cleared the grounds from all remaining doubt. The problem with Dreyfus' or other similar approaches did not lay in the inadequate understanding of the game or its phenomenological analysis, but in

i) treating all AI approaches as necessarily mimetic,
ii) considering conditions (a)/(e) analysed above as inevitably "binary", i.e. either fulfillable or not, while they are "analog", i.e. subject to degrees and therefore to cumulative satisfaction, and hence
iii) undervaluing the fact that, in each chess game, the whole context, elements, facts, rules, memory of the past moves, problems, heuristic strategies and evaluations, etc. make up a logical model which, though extremely vast, is very well-defined (entities, their properties and mutual relations, legal and illegal moves are all fully determined and nothing is left to chance), can be analysed in discrete terms (board games are digital and not analog as ball games) and hence subjected to progressive data processing.

It is true that a Great Master may well have a holistic and visual approach to the game, based on intuition, insights and inexpressible experience, but this is irrelevant to non-mimetic AI, since a computer may play chess better than any human being without even slightly simulating any human process of thinking, very much in the same sense in which we can fly faster than any bird. Likewise, it is true that there are tasks, such as understanding a natural language or translating from one to another, which remain beyond the possibility of a machine, for they are too "open", but chess, being a limited game, is not one of them, and can have its (a)/(e) features subjected to a gradual and increasing fulfilment. In order to play chess decently, a computer cannot compute all possible combinations, since the number of alternatives to be explored is approximately 10^{44} (this is known as combinatorial escalation; to have some clue about the size of the number, you may recall that the number of protons and neutrons taken together in the whole universe is often estimated to be 10^{80}), but it does not need to, for it is usually sufficient to combine three essential operations:

- generate static lists of legal moves, i.e. board positions, restricting the number of moves (the depth of the search, technically known as the size of the search space, is usually five moves, i.e. ten ply, which already generates half a quadrillion moves on the average) to a relative small number that enables it to produce a reply in a reasonable amount of time. There are additional methods to limit the scope of the search and hence improve its depth;
- assess the value of a particular series of moves by means of evaluation functions—that is assign numeric values to each given board position on the basis of a set of features, such as material or positional advantage, strong attack, control of the board, development, etc.—and/or a minimax strategy that attempts to maximise one's gain and minimise the opponent's, and finally
- choose the move that yields the highest value.

The first operation is a function of mere computable power, the second contains heuristic strategies based on what can be learnt from the experience of great players and implemented in terms of rule of thumb, such as ways of evaluating gambits. There are four basic chess values that a computer must consider before deciding on a move: material, position, king safety and tempo. The performance of such operations is obviously subject to improvements through time, at least because they can be supported by a constantly enriched knowledge database of millions of opening games, which have fully codified the first 15/20 best strategic moves for each game, endgames and played games. On the whole, a better "stupefication" of the game is progressively achieved. Once this is combined with a dramatic improvement in the hardware—Deep Blue is a RISC System/6000 Scalable Power parallel System (SP) high-performance computer—and an obvious increase in the machine's capacities to calculate variations—the system contains 256 chess-specific processors working in parallel to partition and solve a chess-board problem, each one capable of searching two to three million positions per second, so that Deep Blue can calculate 50 to 100 billion moves within 3 minutes—it becomes perfectly reasonable to expect that computers will soon be undefeatable. Of course, this does not mean that we shall stop playing chess, for after all we still play a fully computable game such as draughts, nor does it make *Deep Blue* any more intelligent than ELIZA. Here is a nice quotation from the people who constructed it:

> Does Deep Blue use artificial intelligence? The short answer is No. Earlier computer designs that tried to mimic human thinking haven't been very good at it. No formula exists for intuition. So Deep Blue's designers have gone 'back to the future'. Deep Blue relies more on computational power and a simpler search and evaluation function. The long answer is No. 'Artificial Intelligence' is more successful in science fiction than it is here on earth, and you don't have to be Isaac Asimov to know why it's hard to design a machine to mimic a process we don't understand very well to begin with. How we think is a question without an answer. Deep Blue could never be a HAL-2000 (the prescient, renegade computer in Stanley Kubrik's *2001*) if it tried. Nor would it occur to Deep Blue to "try". Its strengths are the strengths of a machine. It has more chess information to work with than any other computer, and all but a few chess masters. It never forgets or gets distracted. And it's orders of magnitude better at processing the information at hand than anything yet devised for the purpose. 'There is no psychology at work' in Deep Blue, says IBM research scientist Murray Campbell. Nor does Deep Blue 'learn' its opponent as it plays. Instead, it operates much like a turbocharged 'expert system', drawing on vast resources of stored information (for example, a database of opening games played by grandmasters over the last 100 years) and then calculating the most appropriate response to an opponents move. Deep Blue is stunningly effective at solving chess problems, but it is less 'intelligent' than the stupidest person. It doesn't think, it reacts. And that's where Garry Kasparov sees his advantage. Deep Blue applies brute force aplenty, but the "intelligence" is the old-fashioned kind. Think about the 100 years of grandmaster games. Kasparov

> isn't playing a computer, he's playing the ghosts of grandmasters past. That Deep Blue can organise such a storehouse of knowledge—and apply it on the fly to the ever-changing complexities on the chessboard—is what makes this particular heap of silicon an arrow pointing to the future. (from: http://www.chess.ibm.park.org/deep/blue/faqs.html#ai)

If a comparison is in order then, we may borrow a term from the military context and say that Deep Blue is only "smarter" than ELIZA is because it can stupefy more. As a test of the value of our interpretation it is reassuring to find John McCarthy, one of the fathers of AI, complain that Deep Blue has very little to do with strong AI. I would add that it is precisely in this distance that lies the condition of possibility of its success.

9. The Success of LAI

Defenders of GOFAI sometimes argue that the threshold of intelligence is constantly and unfairly retreated whenever a machine is shown to be able to reach it, yet it should be clear now that the complaint is unjustified. It is not that some intelligent tasks, like playing chess, no longer enjoy their intellectual status, but rather that they are discovered to be *also* amenable of successful computational treatment. As a human being, one will always need some intelligence to play chess well, but as a machine, one can spare it completely. The possibility of a LAI treatment does not diminish the value of a task, it enriches it of a new dimension. The success in game playing shows that a better philosophy also means better results. A program is found to have more chances to succeed if the problem it deals with is carefully defined, the tasks it performs are reduced to sequences of special-purpose heuristic procedures and repetitive instructions, and the context of application is severely restricted to a sufficiently small possible-world, consisting of stereotypical situations and standard cases, that is to a microworld in which relevance, significance and utterly total novelties are either absent, successfully constrained or fully pre-determined by the programmer. Thus, in each of the ten fields listed in section four, LAI approaches the problem of devising smart machines by attempting to improve one or more of the five (a)/(e) conditions. LAI projects aim to extend the scope and sophistication of tasks amenable to computation by means of fuzzy logic systems, artificial neural networks, parallel computing and quantum computing; to extend the number of knowledge and practical-experience sensitive tasks performable by a computer by means of knowledge engineering and expert systems; and to extend the number of body-sensitive tasks performable by a computer by means of robotics. Combinations of such solutions—together with the development and improvement of new algorithms and the elaboration of increasingly large database, which provide a computer with enough information about the task-environment so that the performance of the task itself becomes as context-free as possible (i.e. the computer is able to construct a model of everything which is relevant)—may help to devise *ad hoc* machines or programs that can cope successfully with an increasing variety of well-specified context-sensitive tasks. In this chapter we shall have a brief look at each field, focusing on some of their features that are conceptually most interesting.

9.1. Fuzzy logic systems

Fuzzy logic (FL) can be interpreted as a superset of Boolean logic (BL). It is common to explain this relation by saying that BL associates only 2 truth-values to any wff (well-formed formula) P(x), i.e. {0,1}, so that in BL any wff has a discrete or "crisp" alethic form, i.e. it is either completely false or completely true, depending on whether or not x satisfies P, whereas

FL employs the concept of "partial truths" to establish how far x satisfies P, that is continuous or fuzzy truth values between the Boolean 0 and 1 (note that such values should not be taken for probabilities, since they do not need to sum to one, have no forecasting significance and do not refer to the statistical strength of evidence supporting the assertion that x satisfies the property P). Although such terminology is slightly misleading, the distinction is intuitive. We can now make it a little more precise by referring to a few basic notions of set theory.

Suppose Σ is a non-empty set ($\Sigma \neq \{\}$) whose members $\{x_1,...,x_n\}$ need to be assessed according to whether or not they satisfy a property P. In BL, we can construct a new set in terms of a mapping $\Delta_{BL}: \Sigma \mapsto \{0, 1\}$ from members of Σ to members of the set $\{0,1\}$. The mapping is formally defined as a function $\mu_{BL}(x): \Sigma \mapsto \{0,1\}$, which is called the membership function of Δ_{BL}, and this clarifies why one often speaks of the function or of the resulting set interchangeably. The function $\mu_{BL}(x)$ generates a set Δ_{BL} of ordered pairs $<x, y>$, such that

1. Δ_{BL} is a subset of the Cartesian product $\Sigma \times \{0,1\}$, which satisfies the property of being equipotent with respect to Σ (there is one and only one ordered pair in Δ_{BL} for each element of Σ and vice versa)
2. $\forall x \forall <x, y>(x \in <x, y> \rightarrow x \in \Sigma)$, the formula means that every x in any $<x, y>$ is a member of Σ
3. $\forall y \forall <x, y>(y \in <x, y> \rightarrow y \in \{0,1\})$
4. 0 = non-membership
5. 1 = membership
6. $y = 0 \leftrightarrow \neg P(x)$, that is y = 0 if and only if x does not satisfy the property P
7. $y = 1 \leftrightarrow P(x)$.

The set $\Delta_{BL} = \{(x, \mu_{BL}(x)), x \in \Sigma\}$ is a Boolean set, out of which we can construct the set consisting of all interpretations of x in any $<x, y> \in \Delta_{BL}$ which are not associated with y = 0 or, which is equivalent (and this shows the binary nature of BL), that are associated with y = 1. Γ_{BL} is a "crisp" subset of Σ ($\Gamma_{BL} \subseteq \Sigma$) consisting of all members of Σ which satisfy the property P. Obviously, the truth-value of the statement $x \in \Gamma_{BL}$ is equivalent to the truth-value of P(x) and can be established by checking each member of Δ_{BL}. The statement is true if the second element of the ordered pair is 1, and the statement is false if it is 0. A similar analysis can now be extended to FL.

Suppose that $\Sigma \neq \{\}$, $\Sigma = \{x_1,...,x_n\}$ and that each x in $\{x_1,...,x_n\}$ needs to be assessed according to the *degree* in which it satisfies the property P. In FL, we can construct a new "fuzzy" set in terms of a mapping $\Delta_{FL}: \Sigma \mapsto [0,1]$ from members of Σ to members of the set of some numeric values in the range of real numbers [0.0, 1.0]. On the basis of the membership function $\mu_{FL}(x): \Sigma \mapsto [0,1]$, $\mu_{FL}(x)$ generates a set Δ_{FL} of ordered pairs $<x, y>$ that is similar to Δ_{BL} as far as properties 2,4,5,6 above are concerned, but now every y in $<x,y>$ can be a real value from the closed interval [0,1], so that:

1. Δ_{FL} is a subset of the Cartesian product $\Sigma \times [0,1]$ which may or (more often) may not be equipotent with respect to Σ (Σ is referred to as the universe of discourse for the fuzzy subset Δ).

Note that, although y in $<x, y>$ can take any value in the range of real numbers [0.0, 1.0], this does not mean that, given a specific model for FL, there is actually an infinite number of interpretations of y available, since each system always sets up a finite range of values for y approximate to n decimals.

Formula (3) for BL becomes now

3. $\forall y \forall <x, y>(y \in <x, y> \rightarrow y \in [0,1])$

and formula (7) for BL becomes now

7. $(y = 1 \rightarrow P(x)) \wedge (P(x) \rightarrow 1 \geq y > 0)$ or, more simply, $1 \geq y > 0 \leftrightarrow P(x)$.

This means that, in FL, the two values 0/1 are only the low and upper bound of the set, so while they are still used to represent "complete" non-membership/membership, a selection of values in between is used to represent what are in fact degrees of possession of a certain property, but may be called, for the sake of simplicity, "intermediate degrees of membership" or "degrees of truth". No actual "fuzzification" of truth is in question, however, and this becomes clear once we realise that

- contrary to what is sometimes stated in the literature, neither Δ_{BL} nor Δ_{FL} is a proper subset of Σ; only $\Gamma_{BL} \subseteq \Sigma$ and $\Gamma_{FL} \subseteq \Sigma$
- while in BL both Γ_{BL} and Δ_{BL} are "crisp", and hence we can use them interchangeably, in FL only Δ_{FL} may be considered fuzzy and only in the sense of having been constructed by means of a mapping from members of Σ to a selection of real values from 0 to 1 whenever the cardinality of the selection > 2
- strictly speaking, and hence more correctly, even in FL Γ_{FL} remains crisp, since it is still the case that x is or is not a member of Γ, no matter to what degree x does satisfy the property P. Γ_{FL} consists of all interpretations of every x in any <x, y> which are not associated with $y = 0$, and the non-binary but multivalued nature of FL is shown by the fact that, contrary to what happens in BL, this is not equivalent to saying that Γ_{FL} consists of all interpretations of every x in any <x, y> which are associated with $y = 1$. It is still the case, however, that Γ_{BL} consists of all members of Σ which satisfy, to some degree greater than 0, the property P. Obviously, the truth-value of the statement $x \in \Gamma_{BL}$ is still equivalent to the truth-value of P(x) and can be established by checking each member of Δ_{BL}. The statement is true whenever $0 < y \leq 1$, and false if $y = 0$.

To define FL as the logic underlying modes of reasoning which are approximate rather than exact, to assume that the importance of fuzzy logic derives from the fact that most modes of human reasoning, and especially common sense reasoning, are intrinsically approximate, and to declare that BL cannot satisfactorily describe "humanistic" problems, all this means to have misunderstood both logics. It is natural to translate "it is true that the flower is fairly red" into "it is fairly true that the flower is red" because, working with a natural language in an informal context, it is often unclear to what degree a flower satisfies the property of being red, but in a formal language and a mathematical domain capable of sharp precision, the former equivalence is a syntactic fallacy, which simply misunderstands the scope of the adverb. Hence, in a fuzzy set an object can have numerous different membership values or "grades", but it is only for the sake of simplicity that we speak of degrees of truth or membership when referring to the alethic values of P(x). What is fuzzy is not the alethic value of P(x), but the property P, which is satisfied by x to a y degree that can be made as precise as one needs. To have a less mathematical analysis, you may try to think of degrees of falsehood. You will soon perceive the absurdity of the whole idea of a Negative FL.

Since in FL the truth-functional values of P(x) give the precise degree up to which P is satisfied by x, for any wff φ it is possible to generate a formal calculus by adopting, for example, the following definitions of the set theoretic operations and logic operators (henceforth I simplify out notation by leaving the specification FL implicit):

- $(\Delta = \{\}) \leftrightarrow \forall x\, (x \in \Sigma \rightarrow \mu(x) = \langle x, 0\rangle)$
- $(\Delta_A = \Delta_B) \leftrightarrow \forall x\, (x \in \Sigma \rightarrow \mu_A(x) = \mu_B(x))$
- $\Delta_A' = 1 - \mu(x)$, which is equivalent to
- $(\neg \varphi) = 1.0 - (\varphi)$
- $\Delta_A \subseteq \Delta_B \leftrightarrow \mu_A(x) \leq \mu_B(x)$

- $(\Delta_C = \Delta_A \cup \Delta_B) \rightarrow (\mu_C(x) = \text{MAX}\ (\mu_A(x), \mu_B(x)))$, which is equivalent to
- $(\phi \vee \psi) = \text{MAX}\ ((\varphi), (\psi))$
- $(\Delta_C = \Delta_A \cap \Delta_B) \rightarrow (\mu_C(x) = \text{MIN}\ (\mu_A(x), \mu_B(x)))$, which is equivalent to
- $(\varphi \wedge \psi) = \text{MIN}\ ((\varphi), (\psi))$
- $(\varphi \rightarrow \psi) = 1.0 - \text{MIN}\ (\varphi, 1.0 - \psi)$

If the truth values are restricted to {0,1} then clearly the truth tables of the special FL in question are exactly as BL's. This is known as the extension principle: any result in BL can be obtained in FL by restricting all fuzzy membership values to {0,1} and FL, as described above, i.e. as a multivalued logic, is interpretable as a generalisation of classical set theory and standard two-valued logic. If more than 2 values are admitted, then in FL some properties of BL, most notably the law of excluded middle, idempotency and distibutivity, acquire different interpretations, depending on the specific axiomatisation adopted. To give a trivial example, to keep BL and FL as close as possible one would have to define a tautology not as any wff which is always true, but rather as any wff which is never false, so that $(\varphi \vee \neg\varphi)$ is a tautology both in BL, where its value is always 1, and in FL, where the value of $\text{MAX}(\varphi, \neg\ \varphi)$ = the value of $\text{MAX}(\varphi, 1.0 - \varphi)$ and the latter is always necessarily greater than 0. This set of "tautologies" in FL is called the support of a fuzzy set.

For a less formal approach, consider now the following case. Suppose we take the universe of discourse Σ to be 10 people in a classroom. The property B = "being born in Oxford" is "crisp" or digital: the set Δ will consist of ten ordered pairs <x, y>, where x is the name of a person in the class and y is either 0 or 1, and Γ will contain only people who are B, i.e. names associated with 1. If we consider now the property C = "living close enough to Oxford", we see immediately that C is necessarily "fuzzy" or analog: the "fuzzy" set Δ will still consist of ten ordered pairs <x, y>, but while x is the name of a person in the class as before, y takes now different real values from 0 or 1, depending on our way of assessing the distance of each person's accommodation from the city centre. The resulting set Γ, however, still consists of names of people who simply satisfy the property C, the difference being that they can now be ordered according to the value of the associated y, i.e. according to the degree in which they satisfy the property C. Oxford University Regulation, for example, states that undergraduates must live within six miles of the city centre, hence our values will range from "distance from the centre > 6 miles" = 0 (if John lives more than 6 miles from the centre he does not satisfy C) to "distance from the centre = 0" = 1 (if Mary lives in the centre she satisfies C completely), with other students living x miles from Oxford for x < 6 all counting as members of Γ, although ordered according to the degree in which they satisfy C.

Fuzzy logic was first introduced by Lotfi Zadeh to treat fuzzy concepts in natural languages. It begun as a typical project in GOFAI, based on the classic mimetic assumptions, and as such it remained for a long time little more than a promising theory. Only once it underwent the usual transformation common to many other GOFAI ideas did it become a successful component of advanced technology. FL applications are becoming increasingly popular especially, but not only, in control systems which deal with inherently analog processes—processes that move through a continuous range of values, generate fuzzy data and cannot be easily modelled by linear analysis—and are usually implemented by means of digital processors capable of dealing only with well-defined numeric values, though "fuzzy" processors are also available. Very schematically, a typical fuzzy system consists of three components or logical stages:

a) input stage

A set of membership functions $\{\mu_1(x),...,\mu_n(x)\}$ makes possible the fuzzification of the source data, i.e. the conversion of a crisp value into a fuzzy value by associating the inputs x with values of y. We have seen that the logical form of each membership function is $\mu_m(x)$: $\Sigma \overset{m}{\mapsto}$ [0,1], where Σ consists now of a collection of source data $\{x_1,...,x_n\}$ provided by input devices such as sensors, switches, thumbwheels, etc., while both the range of values $\{y_{[0,1]1},...,y_{[0,1]n}\}$ between [0,1] to be associated with each member of Σ, and the number of functions (the value of m in $\mu_m(x)$ and $\overset{m}{\mapsto}$) hence of the resulting fuzzy sets, are specified by the system according to previously established requirements. Note that membership functions are often n-dimensional (e.g. a 3-dimensional membership function may map data from sex, age and qualification to profession). The result of the input stage is a "fuzzified" set Δ of ordered pairs $\{<x_1,y_{[0,1]}>,...,<x_n, y_{[0,1]}>\}$. Example: a car antilock braking system (ABS) directed by a fuzzy logic controller (FLC). To achieve a smooth slowing down of the car, the controller needs to provide precise operational instructions to the ABS based on information concerning the nature of the surface (wet, dry, greasy, etc.), the speed of the car (mph), the brake temperature and many other variables in the system. All variables change gradually from one state to the next and in each case the continuous transition cannot be easily governed by means of slightly arbitrary thresholds, which would result in a discontinuous and abrupt change whenever the input value overcame that threshold. So a FLC may be adopted, which defines the input of each variable using some membership function $\mu(x)$ in such a way that the input variable's state, i.e. the y in <x, y>, loses value in one membership function while gaining value in the next. For example, at any one time, the y of the brake temperature will usually be in some degree part of two membership functions: 0.7 nominal and 0.3 warm, or 0.6 nominal and 0.4 cool, and so on.

b) processing stage.

A set (usually dozens) of inferential rules $\{R_1,..., R_n\}$ of the form "IF a certain specified pattern of data occurs THEN perform the specified action", generates another fuzzified set Ω from the fuzzified set Δ. The entire set of rules is usually known as a *rulebase* or *knowledge base* and the program which determines the rule fireability, selects rules for firing and executes the consequent instructions is called an *inference engine*. In each rule, whose logical form is $\Delta \overset{n}{\rightarrow} \Omega_n$, Δ is the antecedent provided by the fuzzified input reached in the input stage (a), and Ω is the consequent, an equally fuzzified output established on the basis of a computation of the values $y_{[0,1]}$ in $\{<x_1,y_{[0,1]}>,...,<x_n, y_{[0,1]}>\}$. Usually, rules have complex antecedents, constructed by connecting several values using the FL operators as defined above. For each n in $\Delta \overset{n}{\rightarrow} \Omega_n$ the inference engine invokes the N relevant rule and generates a specific, fuzzified result usually involving some *linguistic variables*, such as "negative high, negative low, zero, positive low, positive high", for assessment. Hence, the antecedent describes to what degree the rule applies, while the conclusion assigns a membership function to each of one or more output variables. Rules can be solved in parallel in hardware or serially in software. There are several different ways to define the result of a rule, i.e. to determine the value of the output as a function of the rule premise's computed degree of truth, such as the MIN inference method we have seen above when discussing the AND operator, and the PRODUCT inference method. A second stage involves the COMPOSITION of all the fuzzy subsets assigned to each output variable, which are finally combined together to form a single fuzzy set Ω. Rules for COMPOSITION are usually MAX or SUM. We have seen the MAX composition when discussing the OR operator, the SUM composition constructs the combined output by taking the pointwise sum over all of the fuzzy subsets assigned to the output variable by the inference rule. Again, at each passage the appropriate N rules are specified according to previously established requirements. Example: having fuzzified its input, the FLC now elaborates the operative instructions for the ABS on the basis of a set of rules such as: if the brake temperature is WARM and the speed is NOT VERY FAST then the degree in which brake pressure increases is MODERATE.

c) output stage.

In the output stage, the system may need to convert the combined fuzzified results provided by (b) back into crisp variable which is now the representative value corresponding to the specific control value to be implemented. This is known as defuzzification, and it is not always necessary, since there are systems that can cope with fuzzy set. There are many techniques of defuzzification, but two of the most common methods are the CENTROID, which is based on the selection of the centre of gravity of the fuzzy set, and MAXIMUM methods, whereby one of the variable values, at which the fuzzy set has its maximum y, is chosen as the crisp value for the new output variable. Example: the FLC defuzzifies the output of the set of rules to provide a specific "crisp" instruction to the ABS, such as the actual brake pressure expressed by a single numeric value.

Fuzzy control systems are designed empirically, following the usual trial and error methods and adopting precisely that "stupefication" strategy that we have seen to characterise LAI. A detailed analysis and precise documentation of the system's operational specifications, inputs, outputs, rulebase and defuzzification methods (if required) is essential, and is followed by tests. Given its empirical adaptability, fuzzy logic can be fruitfully employed whenever a system is difficult to model, e.g. because it is the result of the experience of a human being, or where data and processes with continuous values are common. Since FL simplifies the design complexity, while often improving control performance, it also reduces the design development cycle and simplifies the implementation of the system (FL controllers can be based on cheap sensors and low-resolution analog-to-digital converters), thus making the latter process less expensive. Moreover, FL systems are intrinsically easily upgradable by adding new rules. All this explains the success of FL controllers. Unfortunately, the latter seems to have spread three erroneous conceptions.

As regarding the philosophy of fuzzy logic, we have already discussed the misleading idea of "degrees of truth" and "degrees of membership": a logic of fuzzy values is not, strictly speaking, a theory that is itself intrinsically fuzzy. As Schroedinger once wrote, "there is a difference between a shaky or out-of-focus photograph and a snapshot of clouds and fog banks". Nor is FL a logic that deals with ambiguity, vagueness or confusion, unless the latter words are superficially used to refer to continuous, degree or analog values as opposed to discrete, binary values. Truly ambiguous, vague or confused information—not only sentences like "perhaps I do not like it as much as I should, or maybe I do, after all", which immediately appear to lack a definite context-free meaning, but also sentences that are semantically or syntactically ambiguous like "when she met us she thought we were going to the pub", which a computer could interpret in 13 different ways—remains so unless it is further regimented and, in case, a fuzzified logic, wrongly understood as a theory of degrees of truth, could only make them *irremediably* ambiguous, vague or confused by isomorphically adapting itself to it. In database jargon this is sometimes called the *GIGO rule* (Garbage In, Garbage Out).

As regarding the philosophy of fuzzy technology, some people seem to believe that FL is the first method that allows the precise and reliable computation of continuous quantities, as if infinitesimal calculus had not been devised. It is true that FL controllers can efficiently replace traditional PID (proportional-integral-derivative) control systems in areas such as consumer electronics, industrial robotics, automation, information systems, diagnosis and other expert systems (DBMS, Information Retrieval), in pattern recognition, in connection with neural networks (Image Processing, Machine Vision) and in decision support systems, but the reason is not because FL controllers allow us to do something utterly new, but rather because PID controllers, though highly reliable and effective, require sets of differential equations to define the system response to its inputs, which are extremely more difficult to formulate. Consequentially, PID systems are very demanding in terms of processing power and memory, may not be as effective as systems based on empirical rules such as FL controllers, and are certainly more expensive and much less flexible and adaptable. It is an obvious case of technological evolution, rather than revolution, and perhaps it would be useful to adopt the logic of fuzziness even when interpreting the development of fuzzy technology.

Finally, in the philosophy of artificial intelligence there seems to be a tendency to attribute FL's success to the alleged fact that somehow it resembles/captures, much better than BL does, the way in which human beings reason, take decision and solve problems on the basis of certain but "approximate" data. It is the old mimetic postulate of GOFAI that often permeates the less technical and more "philosophish" interpretations of FL. Thus, supporters of FL systems may misinterpret the latter in terms of brain/mind-mimetic functions or reality-isomorphic properties, and hence expect FL to be able to solve problems whose intractability does not depend on the kind of mathematical logic employed. Surfing the Internet one may come across views according to which FL will make possible the construction of

> Vast expert decision makers, theoretically able to distil the wisdom of every document ever written. Sex robots with a humanlike repertoire of behaviour, Computers that understand and respond to normal human language. Machines that write interesting novels and screenplays in a selected style, such as Hemingway's. Molecule-sized soldiers of health that will roam the blood-stream, killing cancer cells and slowing the ageing process

or that "the first fuzzy logician was the Buddha". This is nonsense, or better science fiction of the worst kind combined with a pinch of deeply misunderstood spiritualism. It is evidently true that in our minds we rationally conceptualise our experience of the world according to a logic which may often be described as fuzzy—so much so that it is worth formalising such logic into a more precise calculus—but the resulting calculus should not be mistaken for a reliable description of the ways in which we actually perceive the world, think and perform intelligent tasks. To the objection that no human being seems to be conscious of any process of fuzzification, rule-based analysis and defuzzification of the information in her or his possession, the reply can only be that all this must then take place at a sub-conscious level, but the subconscious is a land where only psychoanalysts and metaphysicians dare to fly. It is off-limit to computer scientists, and as epistemologists we should probably be content to acknowledge that, although the logic may change, Boole's hope that it shall one day describe the Laws of Thought is as hard to fade as and mankind's useless efforts to make it come true.

9.2. Artificial Neural Networks

Fuzzy set theory maps elements of an object space into a space of n > 2 membership values. Now, such non-linear mapping can be variously parameterised and can also be fruitfully combined with *artificial neural networks* (ANN).

Although there exist a lot of different approaches to ANN technology, and hence many specific implemented systems that can be very unlike each other, at the core of neural computation we find the concept of distributed, adaptive and non-linear computing so, as a general introduction, it may be sufficient to present an ANN (also called *artificial neural system*, *connectionist architecture*, *parallel distributed processing* and *neuromorphic system*) as a multiprocessor computing system, which can be logically described as a matrix of

- a finite set of individual nodes {PE_1,..., PE_n}, working as processing elements (also known as *units*, *processors* or *neurons*), whose function is to map patterns of inputs into patters of outputs, and
- a finite set of asymmetric (unidirectional) relations {C_1,..., C_n}, working as communication channels connecting members of {PE_1,..., PE_n}, whose function is to enhance or inhibit the transmission of patterns of signals by increasing or decreasing their numeric weight (also called strength). The bigger the weight, the greater the effect a connection has in determining whether or not the relevant PE will fire (be activated).

The processing power of an ANN consists in its capacity to adapt to, or learn from, a set of training patterns by dynamically adjusting the weight of its inter-PE connections on the basis of a non-linear, parallel processing of specified inputs, and according to a pre-established proportional difference between the desired output and the actual output. Depending on how {C_1,...,C_n} is implemented, the topology of an ANN may vary from fully connected to sparsely or just locally connected but, formally speaking, ANNs' graphs usually are not even weakly-connected, i.e. they satisfy the formal property $\neg(\forall x \forall y\ ((x \in \{PE_1,...,Pe_n\} \wedge y \in \{PE_1,...,Pe_n\}) \rightarrow (Cxy \vee Cyx)))$. Hence, ANNs have fixed hierarchical structures: disjoint subsets of {PE_1,...,PE_n} are rigidly ordered into multiple layers from ground-input to top-output level, with inter-layers and infra-layer connections. Since layers are numerable, and their order always includes an input layer IL, $n \geq 0$ hidden layers HL and an output layer OL, and both IL and OL are inevitable, a common way to classify ANNs is by referring to the number of their HLs (usually, a ANN has several HLs).

PEs may have a small amount of local memory, they operate only on local data, each working in parallel with other PEs, usually asynchronously. They may act just as a simple threshold discriminator but, more often, a PE employs a threshold function—typically a sigmoid or logistic function—which makes possible to regulate the PE activation for output, in the following way. Each PE can take many inputs having different weightings but provides only one output which depends on the inputs. When a pattern of signals is received by its input connections with a sufficient weight, a PE "fires", i.e. maps the input modified by the interconnection weight into another pattern of signals, generating a further output and interconnection weight. In order to do so, the PE processes the input and interconnection weight by a summation function (typically a weighted summation) to yield a sum that is passed to a nonlinearity called a transfer function (typically a sigmoid). The output of the nonlinearity is the output of the PE, which is sent, through output connections, to another PE. Hence the operations of each PE are fully determined by the network structure (cardinality and organisation), the connection weights, the inputs received from other PEs belonging to no higher layers via the connections, and finally its own threshold function. The mathematical functions determining the processing behaviour of {PE_1,...,PE_n} are embedded in the network and, although they can vary depending on the specific tasks and needs in question, they are not subject to constant adaptation while the network is working. What make an ANN flexible are the different degrees (which may be fuzzified) of weight that its internal connections are capable to acquire in the course of the mapping process. Thus, ANNs are not programmed to perform a specific task "top down" but "learn by example" (bottom-up organisation) how to process a particular input satisfactorily, before they are employed in an application. To cope with another task, they need to be retrained.

The process of training a neural network involves the following stages:

1. the untrained network is presented with carefully selected patterns of typical input data
2. the network maps the data into an initial pattern of output data

3. the network adjusts the weights of its connections using a variety of functions and according to how much the resulting output patterns differ from what they are expected to be. A training file is developed, consisting of data for each input node and the desired response for each of the network's output nodes. The adjustment of the weights is a matter of trial-and-error, does not follow rigidly programmed instructions and involves no software programming.
4. step (3) is repeated for many typical input patterns, so that the actual output of the network converges with the desired (by the trainer) output.
5. when the gap between actual output and desired output falls below a pre-established (by the trainer) threshold of accuracy, the training process is complete, the network operates satisfactorily and is ready to be used as a predictive or diagnostic tool, to process new selections of the kind of input patterns for which it has bee trained.

Depending on their methods of data processing and training, ANNs can be classified as

- *feedforward* , when they have no feedback and simply associate inputs with outputs.

This type of ANNs is normally used for simple pattern recognition.

- *recurrent*, when they implement feedback relations needed in order to create a dynamic system that will produce the appropriate pattern.

This type of ANNs is normally used for pattern reconstruction.

- *supervised*, when they require a human trainer to tune the ANN to the desired output.

The most widely used *supervised* ANNs are known as Back Propagation ANNs. They are multilayered, feedforward networks which are trained using an error criterion. The network's output is compared with the desired output to produce an error measure; then an algorithm “backpropagates” the error from the output PE to the input PE iteratively in order to adjust the weights increasingly well so as to reduce the error. The network is trained by repeating this process many times. Once the error parameter has decreased below a specified optimal threshold, the network is said to have converged and its training is complete. Back Propagation ANNs are used for classification and prediction tasks. Hopfield networks and Boltzmann networks are special types of multi-layer ANNs characterised by improved algorithmic features.

- *self-organising*, when they can learn to identify structures in the data by adapting automatically in response to particular types of inputs, according to pre-established requirements.

The most widely used *self-organising* ANNs are known as Kohonen and they are networks in which the PEs compete with each other for the “right” to respond to an input pattern. They are trained with unsupervised algorithms that can cluster patterns of data into families on the basis of measured attributes or features (as philosophers, we would say “family resemblance”), serving as inputs to the algorithms. Self-organising ANNs can be used for pattern discovery tasks.

Since ANNs are multiprocessor computer systems, in principle they can process any function computable by a Turing Machine or Von Neumann machine (VNM). Furthermore, we have seen in chapter two that there are mathematical models of ANN that are much more powerful than TMs, the so-called Super Turing Machines, and it has been shown that neural networks can be classified into an infinite hierarchy, depending on their different computing capabilities. However, *in practice*, implemented networks face enormous difficulties when dealing with problems involving symbolic processing and memory resources (e.g. number crunching), while they are much more successfully employed to simplify many computational processes involving patterns of data that would be virtually intractable by ordinary computers.

In particular, networks can be extremely useful whenever large amount of structured data form patterns—not necessarily sensory patterns, such as a picture, but also logical structures, as this is for example the case with statistic or sampling models—which are then available as training input, and when there are classification or mapping needs (clustering and synthesising data, forecasting) that cannot easily be satisfied by the elaboration of computable algorithms or efficient heuristic rules. Since networks "learn" from their examples, they are tolerant of some imprecision and can generate models effectively even from noisy data, without having to discover possible constructive algorithms.

To summarise, the functions and advantages of ANNs are:

1. they can map patterns of data by adapting their structures to input examples
2. they can generalise patterns of data thanks to (1)
3. they can deal with multi-variable, non-linear clusters of data
4. they can deal with noisy, distorted, imprecise and incomplete input data when doing (1)/(3) (but recall the GIGO rule: unclear, uncertain, vague data are not subject to fruitful mapping)
5. they can do (1)/(4) according to specified problem-solving requirements, either by training or automatically (self-organisation)
6. they can require advanced mathematical solutions—at least to determine the state function (e.g. the summation function), the transfer functions (e.g. sigmoid function) and the training algorithm (e.g. back-propagation)—but no software programming to do (1)/(5). It is common to formulate such a property by saying that ANNs do not require algorithms, but we have just seen that this is slightly misleading. ANNs approach problem solving tasks in a different way than software-based computers, which follow sets of instructions and hence require programs in order to perform their computations, but ANNs still require increasingly sophisticated mathematical functions to perform their tasks successfully.
7. they have a relatively short development time because of (6)
8. they can operate at high speed
9. they can be implemented on parallel hardware for even greater speed of operation
10. they can be employed in real time operations because of (8)/(9)
11. they have some fault tolerance and are subject to graceful degradation (a partially damaged network may still perform some of its essential tasks satisfactorily) because of their architecture and the redundancy in their information coding.

There follows that their conditions of applicability are:

a) the problem to be dealt with can be expressed as a mapping problem (necessary)
b) sufficient samples of typical sources of data to be mapped are available (necessary)
c) tolerance for errors and context sensitivity is required
d) the problem allows no computable algorithmic solution or only algorithmic solutions that are too complex or difficult to formulate.

And their areas of application are:

- pattern recognition, detection and reconstruction
- data classification and generalisation
- functional prediction/projection (conclusions are generalised for new patterns from past patterns)
- system modelling of physical processes for which there are no algorithmic interpretations or algorithmic interpretations are too complex and difficult to be found
- control problems, where the input variables are measurements used to control an output actuator, and the network learns the control function.

More practically, some real-world, typical applications of ANN technology include: voice and visual recognition, spectra identification, chemical structures, biomedical instrumentation, medical diagnosis, credit rating (loan risk evaluation), forecasting of future sales, investment analysis, where predictions can be attempted on the basis of patterns provided by past data, market performance, economic indicators, writing recognition (especially signature verification, when new signatures need to be compared with those stored), signal processing and compression, automatic control and monitoring, production requirements, quality control, oil & gas exploration.

The history of ANN has not always been so successful. Its "prehistory" can be identified with cognitive associationism, whose first roots Hobbes thought could be traced back as far as to Aristotle. It is one of the chief, unifying traits of the empiricism's conception of the mind. Simplifying, the mind is conceived as devoid of any innate knowledge but endowed with the capacity of forming ideas by manipulating basic sensory experiences (sensations). At the end of the nineteenth century, the development of what is now called neuropsychology led to a plausible, very general account of how some mental processes might be underpinned by the interactions of neurons in the brain. The brain began to be mapped and functions to be localised. In this context, it became increasingly clear that the behaviour of nerve cells is at least also determined by their chemio-electrical excitation, and that an excited cell can "fire" and thus cause the neurons to which it is connected to become excited as well. It was then further suggested, more controversially, that associations might be formed between two cells when they tend to become excited simultaneously. In 1943, the neurophysiologist Warren McCulloch and the logician Walter Pitts—inspired by Ramon y Cajal's theory of the neurone structure, by Charles Scott Sherrington's theory about the excitatory/inhibitory synapses and by Alan Turing's work on computing machines—proposed an abstract mathematical model of the first artificial PE. Their aim was to discover how a brain might engage in logical reasoning and how this psychological functioning might be explained on a neurophysiological basis. Different configurations of McCulloch-Pitts artificial neurons were found to be able to perform the same logical operations that are carried on by a Turing machine. Later, Frank Rosenblatt developed a model of network of McCulloch-Pitts artificial neurons capable to respond to different patterns, which he called "perceptrons". ANN applications begun to be envisaged since the late 1950's, but 1969, in a famous book entitled *Perceptrons*, Marvin Minsky and Seymour Papert showed that there were a number of specific patterns that Rosenblatt's networks, having no hidden units, could not model. They further claimed that this limitation probably made connectionist networks inappropriate as models of human cognition. There followed more than a decade of stagnation of research in the field. Perceptrons were limited in scope and too demanding in terms of resources even to solve simple pattern recognition problems such as horizontal/vertical line discrimination. Interest in ANN grew again only in the eighties, once parallel architectures and mathematical algorithms became sophisticated enough for general applications. In particular, the recovery of the whole program of research, the so-called Neo-connectionism, was possible thanks to developments in the study of multi-layered networks, an extension of Rosenblatt's original design that can overcome the limitations of the former by means of back-propagation algorithms, formulated in 1986 by David E. Rumelhart and Jay L. McClelland. Their *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* is unanimously recognised as the starting point of the contemporary renewed fortune of ANN technology and applications.

Even this sketchy summary is sufficient to show that ANN technology has achieved its successes only because it has abandoned its initial, mimetic ambitions and has endorsed a purely engineering approach. ANNs are collections of mathematical structures with the ability to modify their format. They can be designed as computer models, but when they are implemented they often are software simulations rather than hardware systems, so that while a BNN is always "a piece of hardware" an ANN could be nothing more than a sub-routine written in C, which typically will have been automatically generated by a neural net development kit. Their design and modelling is driven by and tested against technological aims, and the construction of powerful analytical tools to solve a variety of practical problems is pursued by developing more efficient mathematical algorithms and architectures that can easily be transferred to silicon chips. Interestingly, the more they evolve, the less "biologically plausible" ANNs seem to become. This is not a problem, since Neo-connectionism in computer science is a successful technology that is not concerned with physiological or psychological phenomena in BNNs and should not be confused with either neurophysiological, cognitive or philosophical theories. Of course, connectionism is also a neurobiological theory and a standard model of the brain and its functioning, but the fact that neurobiological connectionism is a successful program of research does not mean that ANNs must have a significant relevance for neurobiology, or that LAI could really benefit from a good understanding of information processing in BNNs. As we shall see in a moment, the more we understand them the more we realise that biological networks have little to do with artificial ones, and if we exclude the handy use of some common terminology—which may also be rather misleading sometimes—it is often remarked that all ANNs have in common with BNNs is a couple on "Ns" in their labels. Understandably, however, successes in computing and neurobiological connectionism have given rise to new hopes for an equally successful, yet old-fashioned "mimetic connectionism" in cognitive science and in the philosophy of mind, and this despite the anti-mimetic orientation of technological Neo-connectionism, the non-technological nature of neurobiological connectionism, and the fact that actual ANNs work increasingly well precisely because they do not try to mimic biological nets anymore. The contradiction is obvious, and attempts to overcome it often leads to a number of popular mistakes. What follows is a small "gallery of errors".

a) What ANNs can do.

In the "pro-mimetic" literature, artificial networks are often claimed to be "biologically plausible" because they show at least the following major "biological features":

1. they can derive meaning from complicated or imprecise data
2. they are capable of abstraction
3. they can formulate prediction about possible future events
4. they are highly adaptive, and can mimic the human brain's most powerful ability, namely that of pattern recognition.

However, none of these anthropomorphic claims is actually supported by the technology in question:

1. ANNs are useful complex tools for mapping patterns of data, but whether the latter are *meaningful* or not, and what their plausible meaning may be, remains up to the human user to establish. This also applies to the *post hoc* discovery of how a network has managed to generate the "meaningful" data it has achieved.
2. ANNs' capacity to discover structural generalisations and similarities between different patterns is sometimes overemphasised in a way which reminds one of Locke's simplistic yet erroneous theory of how universal ideas are "constructed" out of an elementary process of abstraction. Not only a network is not a magic hat out of which one may get utterly new information that is not already present in the training examples—ANNs are not

epistemically creative, and whatever information they can obtain from the data it must be already encoded in the data—but the process of artificial as well as biological abstraction of similar features shared by a set of individual examples, while certainly capable of leading to the construction of a *more general* model (cf. Locke's universal idea), it is always already based on the assumption that all members of the set in question do share such similar features, since we need a well-defined membership rule to construct the initial set. In other words: a network may reconstruct the face of a burglar out of a limited number of snapshots, only if (i) the relevant features of the person are actually captured by the pictures and (ii) the latter are all of the same person. Two silly requirements that are easily forgotten.

3. ANNs can only provide projections from previous data, they do not "interpret" the future.
4. ANNs are highly adaptive only as a *type* of technology but not as *tokens*. Each individual ANN stores "experiential knowledge" of the right way of mapping its patterns isomorphically, i.e. as a particular matrix of PEs and connection weights which represents the "solution" to a problem. Hence, each individual ANN is always a dedicated system, which is capable to map different kinds of patterns of data only if it is completely re-trained, but a new training means the loss of the old format (to have more flexible ANNs the best thing to do is to combine their technology with programming algorithms which can store and retrieve specific formats of "ANN-matrices"). One may think of the famous piece of wax in Descartes' hands. As a kind of substance the wax is highly malleable, meaning that a piece of wax can take any shape we may fancy, yet the same piece of wax can take only one shape per time. Finally, it is plainly wrong to say that pattern recognition is the human brain's most powerful ability, but even if it were, ANNs do not mimic it. To understand why, we need to consider the next two popular errors.

b) ANNs learn as human beings do.

This mistake should already be clear from what has been said above, but let us consider a couple of examples from the standard literature. First case: a 4-inputs PE is trained to fire when the input is 1111, and not to fire when the input is 0000. After applying the generalisation rule the PE will also fire when the input is 0111, 1011, 1101, 1110 or 1111, it will not fire when the input is 0000, 0001, 0010, 0100 or 1000, and when any other input is present, such as 1001, it will produce a random output. Second case: we digitally input a photographic image for a neural network to identify, "guessing" which circuits to fire to recognise the photograph and output the correct answer. In both cases, we train the ANN by strengthening (resistance turned down) the connections between individual PEs when the task is performed correctly and weakening them (resistance turned up) if the task is performed incorrectly. In both cases, the neural network can "learn from its mistakes" and give more accurate output with each repetition of the process. Now, if a person should behave in the way just described we would be very worried indeed, and rightly so for more than one good reason. Firstly, neither a human brain nor a human being needs to be trained necessarily in an *iterative* way to learn to recognise a pattern, e.g. the shape of a new, strangely looking, revolving door in order, to be able to evaluate whether one can pass through it. Secondly, not even Augustine—despite what Wittgenstein writes—held a naive theory of merely ostensive learning. Most of our cognitive organisation of the world surrounding us seems to require much more than mere exposure to spatial patterns, since it is usually associated at least with experiential memory, expectations, interaction, education and language, and is the result of a constructionist conceptualisation. Besides, human experience has the intrinsic propensity to become atomic. Two human beings do not recognise a particular pattern in the same way: one will immediately (non-inferentially) perceive a television screen, the other will immediately perceive a computer screen; one will immediately perceive a Triumph Spitfire 1300, the other only a car. The same individual, trained to recognise different types of cars, will have lost the capacity to see only vehicles. Thirdly, it is odd to be forced to remind to ANNs supporters that a human brain can learn an infinity variety of tasks, including recognising an extraordinary range of patterns, but none of them "erase" the previous ones, yet we have just seen that this is the case with ANNs matrices, whenever a network is trained with a new series of samples. ANNs are often praised for the massive parallelism of their computations, yet lacking short-term memories they are practically unable to do more than one thing at a time. Finally, there is a problem of levels of analysis. It is extremely unclear, not to say very confusing, whether people defending the similarities between artificial and biological learning processes are only limiting themselves to comparisons between an ANN and a brain, i.e. with the way in which neurons and synapses behave, or wish to extend their remarks to how people actually learn to recognise patterns. The often mentioned reference to a child learning to recognise the shape of a dog inclines one to suspect that the latter may be the unfortunate case, but then the absurdity is manifest, since one may learn to recognise the pattern of a graph, for example, by associating it with her favourite shape of cake, a round pie, and neurological, psychological and mental levels, if they are levels and not just independent phenomena, should not be confused. The point is not worth pursuing any further, but if a comparison cannot be avoided all together, then it is just a bit less misleading to compare ANNs training to the way in which photographic film is impressed by light.

c) ANNs closely mimic the internal structure of the brain and they closely simulate its behaviour.

A reply to the previous criticisms usually takes the form of a quick retreat and a counterattack. Artificial networks are indeed different from biological brains because they implement a much more simplified model. Nevertheless, the architecture and the processes are still very much the same, even if on a smaller scale, and by studying the former, we may learn a lot about the latter. This seems an odd muddle of mistakes. To begin with, the contrast between ANNs and BNNs is not a mere matter of scale or magnitude but essential, and current ANNs are not just incredibly simpler than BNNs, they are substantially different. What lies behind the opposite view is a combination of

- some form of functional associationism about how mental contents come to be shaped
- a structural atomism, according to which the whole has at most only amplified properties inherited from its components, and a grasping of the nature of the most elementary components will lead to a reconstruction of the properties of the whole, and
- a methodological modularity, according to which the whole can be reconstructed piecemeal by a progressive combination of elementary components, and therefore functions that can be implemented correctly in a restricted domain can also be gradually enhanced until the domain eventually approaches the real-world in complexity and other features.

But as the former tenet is too naive, so the latter two are exceedingly simplistic. A complex ANN may consist of hundreds, perhaps even thousands, of PE and connections, but compared to a brain it has the complexity of a grain of sand, and it is hard to see how such an impoverished type of knowledge representation system may give rise to higher-order concepts or intellectual activities. The brain consists of about one trillion (10^{12} = 1 thousand billions) neurons and a network of a thousand trillion (10^{15}) of synapses (these are the ending parts of the axons). Now numbers here are not mere factors which allow us to grasp the nature of a BNN by simply multiplying by billions the features we find in an ANN. "Numerical advantage" is only another way of stressing the fact that systems have emergent as well as "submergent" properties: the former are not present in their parts but appear more and more distinctly only in whole combinations of them, the latter are present only in the parts and gradually disappear when the size of their bearers is scaled up, so "bigger" is usually equivalent to "different". These emergent/submergent properties often represent what is really interesting in the system. The rich texture of a pencil stroke, the sponginess and taste of a cake, or the strength of an army are just a few simple cases of emergent properties; the lightness of a pebble, the brittleness of a sheet of paper and the actual computability of a problem are examples of submergent properties. Now, nobody would suggest that we may assess the strength of an army, for example, only by observing the strength of a single platoon, even if the platoon was exemplary, nor would anybody think of attributing to the whole army the mobility of a platoon. Yet, in the case of the present comparison, to look at ANNs as just smaller models of BNNs is like trying to reconstruct and understand the whole military history of World War II (mind that the relation is not yet correct, as there were only a few tens of millions of soldiers fighting) by looking at the behaviour of a platoon in a pacific island. Not only this is not a merely quantitative problem, it is above all a matter of completely different properties. An atomist and modular approach in the philosophy of mind may then lead to a few initial successes concerning some elementary functions, typically in the area of conscious perceptual experience, but is bound to fail when more general explanations are soon required. As Dreyfus has written, climbing a tree is not a first successful step towards reaching the moon, it is the end of the story.

Secondly, although the life of the brain's biochemical universe is still largely unexplored, we do know not only that neurons can promote, amplify, block, inhibit, and attenuate the micro-chemio-electric signals which are passed on to them and through them, somehow like PE, but also that, contrary to PE, neurons are self-repairing and self-wiring, that there are many neurotransmitters of several different types (chemical agents that can transmit the activity of peripheral nerves onto their target organs) whose chemical properties, not just electrical intensity, deeply affect the behaviour of neurons; that the threshold membrane potentials in real neural networks are not completely plastic; and that they are probably not homogeneous processors. It turns out that even the platoon we are talking about belongs to a completely different army. The objection is often that, no matter how macroscopic the differences may be, after all we are still dealing with the same logical models, and it is at this abstract level of description that similarities between ANNs and BNNs can and should be appreciated. The best that can be replied is that nobody denies that, historically, neural computing was loosely inspired by investigations into modelling nervous system learning, nor indeed that a simplistic and rather superficial analogy concerning the very general structure of both ANNs and BNNs can be drawn. Like the brain, we have seen that ANNs can be thought of as consisting of inter-connected neurons linked together by synapses and capable of firing electrochemical signals. What needs recognition is the crucial fact that this is as far as the analogy is actually worth pursuing, especially once we realise that, with a bit of effort, a logical model of London's traffic lights system could be provided on the same basis. Some people seem to forget that there is a very wide variety of different types of ANNs, and that if a model, and not just one particular type of ANN, must be elected as the proper interpretation of the brain (is the brain a Kohonen net, a simple Back-propagation network, a Hopfield network, a Boltzmann network or what else?), then its logical description needs to be so generic to become, in the end, uninformative.

d) Symbolic vs. Connectionist approach is an ultimate alternative.

The success of ANNs is often explained on the basis of a philosophical assumption: if ordinary VNMs are not able to perform some intelligent tasks, most notably pattern recognition, this is because they lack the kind of structure and functioning we find in biological neural networks. By reproducing more closely the physical architecture of the brain—it is further argued—it will become possible to simulate brain functions increasingly well. From what has been said so far, it should be clear that such a view is mistaken because ANNs are not sufficiently similar to BNNs and because pattern recognition can be done successfully on conventional computers too, although writing the software is more difficult and time consuming, and the system will generally have a slower response. The symbolic approach—sometimes also called "classic", according to which the brain is just a physical symbol system, i.e. a mechanism for interpreting and manipulating formal representations—and the connectionist approach to GOFAI competed with each other for some time only because they were both mimetic, i.e. they both tried to device the ultimate, unified theory of the brain, a goal that admitted only one winner. One may recall, for example, that the aim of McCulloch and Pitts was to show that an ANN could implement a Turing machine. But we know that neither a VNM nor an artificial neural network is necessarily meant to be a model of the brain or simulates closely enough its functioning, and that symbolic, software-based computers and connectionist, artificial neural networks offer only different approaches to computing that, contrary to popular belief, are not in competition but complement each other. So much so that ANNs are normally implemented on conventional computers and the latters are often used to supervise the neural network in order to perform at maximum efficiency. It turns out that a statement such as "artificial networks are better models of the brain than conventional algorithmic computers" is not very helpful, for there is no such an alternative. Both approaches can share the questionable, mimetic view that the brain is a special instantiation of a computing machine.

Much more could be said on the topic, but I hope that the previous discussion has sufficiently clarified the main issues. There is no good reason to believe that ANNs and BNNs have necessarily anything more to share than a general conceptualisation of their models. The construction of ANNs was *inspired* by the discovery of the densely interconnected, parallel structure of the brain, and the construction of *neuromorphic systems* was an actual program of research when GOFAI was still thought to be possible. It failed. Today, actual ANNs do not work successfully *because* they mimic the brain's structure and functioning, and the context of discovery should not be confused with the context of justification, nor taken together with the technological success of ANNs as a good reason to defend "mimetic connectionism" in philosophy or in cognitive science. The construction of ANNs does not need and can perfectly afford to ignore any further "neural inspiration". As for "cognitive connectionism", in so far as it supports the general view that cognitive functions depend, as emerging properties, on neuro-chemical events distributed throughout the brain in processing elements linked by neural connections, its position is perfectly safe, if perhaps philosophically uninteresting. But when cognitive connectionism moves further, and attempts to reduce higher order mental states, including consciousness and feelings, to such neural distribution, modelling the "sub-conceptual" level of description of cognitive processes using methods offered by technological ANNs, then it loses all neurological realism and much of its scientific value, opening the path to all possible speculations. ANNs do not resemble BNNs closely enough to attribute to such an isomorphism their successful performances, and technological interesting applications will more easily escape out attention as long as we shall waste our resources trying to clone an electronic model of the brain to simulate its intelligent behaviour.

9.3. Parallel Computing

ANN technology can be interpreted as a branch of parallel computing. As such, it is hardly the most successful, for the alternative, represented by CPU-based applications, is the most promising development of computing in the close future.

Suppose you are in a magic castle. All rooms have one-way doors that allow you to move only in one direction. There is a treasure in the castle, and your intelligent task is to discover it. It is a search problem (where is the treasure?) which can easily be transformed into a decision one (is the treasure in room 1? Is it in room 2? etc.). The solution can be achieved by going through each room (the problem space) systematically. You would be acting as a *vector*. You may be lucky, and find the treasure in one of the first few rooms you visit, or indeed the magic castle may be so big that you shall never enter the right room. How can you improve your chances to discover what you are looking for? The options are few:

1. run as fast as possible from one room to the other
2. improve the direction of your searches, e.g. walking through the corridors looking inside the rooms without entering into them
3. ask Merlin to scale down the castle and make it as well-structured as possible. The number of rooms remains the same, but they are now small and placed in neat rows, so options 1 and 2 become even more efficient.

Of course, the system castle/treasure/hunter is just a very simplified analogy of a classic computer. We have seen that VNMs have remained conceptually the same since the first were invented. The real difference is made by our abilities to exploit their computational powers more and more successfully. Present computers have faster clock, use better algorithms and have increasingly efficient architectures, while nanotechnology has greatly reduced the measure of their circuits. Is this an endless process, that will allow us to find any treasure more and more quickly, or is there something that shall escape our search for ever? Some reasons for pessimism are evident. There are physical boundaries to how many logic elements can be packed into smaller and smaller volumes and clocked at higher and higher frequencies, and strategies (1) and (3) face the physical constraints represented by the speed of light and atomic extension (the value of research in optical computers—standard VNMs capable to process very fast pulses of light instead of slower pulses of electricity—can better be appreciated from this perspective). Indeed, one may reckon that our technology is not very far from running against them. Strategy (2), however, offers a different perspective. Software development is largely a matter of human ingenuity and there is no theoretical boundary to the number of new algorithms that can be devised, or to the extent they can be further improved. This explains why (2) can also lead to a fourth solution of our treasure-problem:

4. you may plan your search by organising a team of treasure hunters, e.g. by assigning a floor to each one of them.

This is known as parallel computing, and it is made possible by algorithms and procedures that can efficiently network and orchestrate multiple processors to execute parts of the same program simultaneously. The conceptual strategy is "Fordist" in nature, hence simple and powerful: the physical constraints, to which the computational power and the maximum execution speed of a machine are subject, can partly be overcome if long and difficult instructions, constituting a single computing task (vectorial analysis), are broken down into smaller and easier instructions, which can then be more quickly executed independently from each other by a series of collaborating PEs, working simultaneously as components of a single *parallel processing computer* (PPC).

A common way to introduce a simple model of a PPC is by considering it a multicomputer, based on the Von Neumann model of computation—a number of CPUs, connected together into a network, access instructions and data from their own memories and execute different sub-tasks serially—or an abstract model of a parallel multiprocessor, called PRAM (Parallel Random Access Machine), in which identical processing units can access any element of a commonly shared memory in the same amount of time. More generally, we may say that a PPC consists of

1. a set (from hundreds to thousands) of central processing units (often termed, in this case too, PE, processing elements) with a high computational rate, using a large and fast memory (either local or shared)
2. high speed interconnection network and/or bus connections linking the PEs. Traffic among the PEs may be more or less intense, since, depending on the architecture, the PEs may run independently of each other, possibly under the control of another processor, or may need to co-ordinate themselves constantly, and
3. special algorithms (often written in Fortran77, C, and C++) required to carry on the organisation and synchronisation of "parallel tasks", i.e. the temporal co-ordination and execution of the logically discrete sections constituting the computational work.

If any more "intelligent" tasks will be performable by our future computers this is very likely because of our improved capacities to make them work in teams. What cannot be sufficiently stupefied to become manageable by a single Von Neumann machine may be a joke for an army of them. And yet, a word of warning is in order. For it would be a mistake to assume that nothing can resist the "parallel force" of a PPC. We shall return to this problem in the next section. Here, we need to recall that, when evaluating the advantages of parallel computing, four essential factors must be taken into account, and these are sufficient to show the limits of any PPC.

i) The appropriate nature of the problem to be solved.

Not all computational problems are "parallelisable", i.e. suitable to logical partition of tasks, or such that they can really benefit from it. Problems that either lack a structured solution or have only a vectorial solution (a concatenated solution with a serial structure in which there is a rigid, linear dependence between tasks, whose order of execution affects the results of the overall computation), present no logical features that a parallel system can fruitfully exploit. They can still be processed by a PPC, but their relative speedup (the ratio between real time of serial execution and real time of parallel execution) is equal to 1. Compare two typical examples: the computation of the potential energy for each of several thousand independent conformations of a molecule and then of the minimum energy conformation is a parallelisable problem; the computation of the Fibonacci series (1,1,2,3,5,8,13,21,...) by use of the formulae $F_0 = 0$; $F_1 = 1$; $F_{n+2} = F_{n+1} + F_n$, for $n \geq 0$ allows only a linear approach. In this case, a PPC would be no more useful than a standard computer.

ii) The appropriate nature of the parallel architecture used to treat (i)

Suppose we are dealing with a fruitfully parallelisable problem, such as our treasure decision problem. What kind of parallel architecture is best suited to treat it? Parallel computing functions can be implemented in several ways. Flynn's classical taxonomy—based on the combination of the number of instructions, which can be executed at a time, and the number of data (data streams) on which a computer can operate at a time—distinguishes between four classes:

1. SISD (Single Instruction and Single Data Stream)

This is the traditional model of a sequential computer (VNM), such as the simple PC discussed in chapter two, in which instructions are executed one by one, on a unique series of data retrieved from memory. Note that if a SISD machine needs to perform the same instruction on many different data, then it must fetch the same instruction many times, once for each datum. Things get even worse when there are loops that force the machine to execute many instructions for each piece of data. Because of this architecture, an ordinary VNM can perform much more slowly than the arithmetic unit is capable.

2. SIMD (Single Instruction and Multiple Data Streams)

This multiprocessor machine is one of the typical models of a PPC: all PEs execute the same instruction (search for the treasure) synchronously, but on their own data (in different series of rooms) retrieved from their own local memory, under the control of a supervising CPU.

3. MIMD (Multiple Instruction and Multiple Data Stream)

This is the typical multiprocessor model of a PPC, in which synchronisation is no longer a necessary requirement, and each PE can execute different instructions on their own data, and communicate asynchronously whenever needed.

4. MISD (Multiple Instruction and Single Data Stream).

This is not an implemented model of computation.

Another standard classification of parallel systems, based on memory and communication models, results orthogonal to Flynn's:

a) SM (Shared Memory) system, in which multiple PEs access and use the same global memory, communicating very rapidly via common memory locations, so that a PE writes to a location, and the other reads from it.
b) DM (Distributed Memory) system, in which each PE has its own local memory, and communicate (more slowly than in (a)) with other PEs by creating and sending data packets (messages) which are received and used by another PEs. DM systems are therefore also known as "distributed memory, message passing systems".

Combinations of (2)/(3) and (a)/(b) give rise to hybrid models and, more significantly, to shared memory MIMD systems and distributed memory MIMD systems. Now, different parallel architectures may be more or less appropriate to solve different parallel problems, depending on the "granularity" of the latter, i.e. the amount of computation that can be done in parallel tasks. Granularity of parallelism ranges from fine to coarse. If the granularity is fine, then the latter has a high degree of modularity (its solution can be reduced to a large but still finite series of sub-tasks, each of them rather small in terms of number of instructions and execution time) and requires a low degree of distributed computational power (only a limited amount of computational power is needed to carry on each sub-task); if the granularity is coarse, then the degree of modularity of the problem is low and the required degree of distributed computational power is high (the solution can be reduced to a rather small series of sub-tasks, and each of them can be solved concurrently but using a large amount of computing power). Clearly, the more finely grained a problem is the greater the potential for parallelism and hence speed-up are and the better it can be computed by means of a SIMD system, whereas the more coarsely grained it is, the more a MIMD system will be suited. However, the finer the granularity, the greater the parallel overheads (time wasted in running parallel tasks as opposed to execute computational instructions) of I/O tasks, synchronisation and communication latency (time wasted by a PE waiting for messages to be sent/received; this is not to be confused with bandwidth communication overhead, which refers to the rate at which data are exchanged between PEs), while risks of other serial bottlenecks (chains of instructions that must necessarily be executed one after the other), negative load imbalance (an uneven distribution of work among parallel tasks resulting in some PEs being idle while waiting to begin the next computational instructions) and even deadlocks (all tasks waiting to receive data, but no one

sending them) increase.
iii) The specific elaboration of appropriate algorithms to match (i) and (ii).
Even if a problem can be successfully parallelised by exploiting the right architecture, parallel methods still require the elaboration of specifically designed algorithms. Everyone is acquainted with the difficulties inherent in co-operative problem solving, which usually need a clear assignation of tasks and a rather firm division, if not a hierarchy, of responsibilities. Likewise, detailed and precise instructions for partitioning the overall problem into separate tasks and allocating tasks to processors are vital. Methodologically, parallel software should then satisfy four conditions:

- modularity (the partition of complex tasks into more simple sub-tasks components)
- high concurrency (capacity of parallel multitasking)
- high scalability (capacity to maintain a satisfactory performance when the number of processors involved increases), and
- (possibly) high locality (maximum reduction of time spent in send/receive processes in favour of read/write processes), since access to data stored in local (same PE) memory is less time-consuming than access to data stored in remote (different PE) memory.

iv) The efficiency of the parallelisation (low degree of overhead) and the upper limit of parallelisation.
There are limits to how far a problem can be fruitfully treated by a multiprocessor system and the boundary of efficient parallelisation is well formulated by Amdahl's Law (named after Gene Amdahl): if 1.0 denotes the whole computational task, and S is the fraction of the computation task that is intrinsically sequential (at least the unparallelisable I/O stage), then $1.0 - S$ is the fraction of the computation that can be parallelised, and the ideal, maximum degree of speedup achievable using N processors is given by the ratio $1.0/(S + (1.0 - S)/N)$. Of course, Amdahl's Law ignores all other real-life factors, but it is already useful to clarify two facts. Firstly, the speedup tends, but can never be equal to 10, the more processors we employ in our system, since every computational problem has a sequential component that will eventually limit the speedup of its parallelised version. Secondly, and more interestingly, there is an optimum threshold of costs/benefits beyond which adding more processors is no longer worthwhile. For example, assuming that the problem is 90% parallelisable, that $S = 0.1$, and that a system has

- 5 processors, we will have an ideal speedup = $1.0/(0.1 + (1.0 - S)/5)$ = ca. 3.57
- 50 processors, we will have an ideal speedup = $1.0/(0.1 + (1.0 - S)/50)$ = ca. 8.47
- 500 processors, we will have an ideal speedup = $1.0/(0.1 + (1.0 - S)/500)$ = ca. 9.82.

Clearly, the effective jump is from 5 to 50 processors, while adding another 450 processors is exceedingly expensive and does not speed up the performance of the system significantly. More technically, the slope of the function can be directly related to the degree of scalability of a parallel system, to which the addition of more processors (additional CPUs are added to absorb the increased transaction volume or more complex processes) yields a proportionate increase in parallel speed-up. More trivially, two people working on the same task are much better than one, but three may not make a big difference, while four may be a hindrance.

In the short history of computing, parallel technology is a relatively new phenomenon. In 1982 two Cray-1 supercomputers linked in parallel (the Cray X-MP) proved to be three times faster than a single Cray-1, and the year after Thinking Machines Corp. and Ncube begun to promote parallel processing as a commercially viable solution. Because of their power, PPCs are also sometimes known as *supercomputers* while they have also been denoted by the expression *high performance computers* because of the HPCC program (High Performance Computing and Communications) launched by the US Government in 1991, although in both cases the expressions included other systems not necessarily based on parallel architectures. In 1997, only one of the top twenty fastest supercomputers relied on a vectorial approach, all the others were parallel processors. This advancement in parallel computing caused the USA to become once again the leading country in a sector which had been dominated by the Japanese industry since 1990: sixteen of the supercomputers in question were built by American companies, only four in Japan.

This short history does not mean that the advantages of a PPC are either marginal or unclear. On the contrary, given the same computational power, it is already becoming more cost-effective to produce a PPC using a number of fairly fast but inexpensive processors, whose cumulative memory can also be very large, than a PC based on the last Intel model. Interestingly, dual-processor capability was already a required feature for entry-level workstations during the first quarter of 98. A dual-processor PC has a motherboard that contains two CPUs and is designed to run an OS such as Windows NT, which supports symmetric multiprocessing (SMP). SMP is a multiprocessing, scalable architecture in which two or more CPUs, connected via a high-bandwidth, share the same memory and hence the same programs, and are managed by the same OS in a roughly non-hierarchical way ("roughly" because one of the CPUs is still responsible for the booting procedures). The presence of more than one CPU may improve fault resilience—the system may still be able to work with only one processor available—and obviously boosts the performance of the system, since applications can be run by the several processors interchangeably, and all multithreaded applications, multitasking or concurrent operations are parallelisable and can be executed simultaneously. This is not yet full parallelism, for software is not specially written for multiprocessors computers, but it certainly represents an interesting step towards it.

In a world more and more pervaded by digital networks, it is to be hoped that PPCs will be able to take advantage of the growing distributed computational power that is becoming increasingly available. It is already possible to device a networked computer as a PPC, rather than just a dumb terminal. Distribute computing and the possibility to concentrate computational resources on important computational problems is one of the next crucial challenges of ICT. PPCs address the possibility of scaling up some computational problems (solving bigger problems in the same time) rather than just speeding them up (solving the same problems in less time). This means widening the possibilities for computational applications. I remarked in chapter one that numerical models and simulations are a fundamental complement to theory and experimentation in engineering and scientific research and have become crucial also for the development of entertainment products. Computational inquiry is more and more often employed in the study or reproduction of all sorts of phenomena that are either too complex (e.g. because of the time scale), expensive or dangerous to be dealt with directly, through realistic physical experiments. The number of fields of application is constantly increasing, and include what have been identified, within the HPCC project, as the "Grand Challenges", fundamental problems in science and engineering, with potentially broad social, political, and scientific impact, that can be advanced by applying high performance (including massively parallel) computing resources:

1. Computational Chemistry (CC)—parallelise key chemistry codes that permit researchers to study environment problems, using techniques such as self-consistent field (SCF), second order many-body perturbation theory (MP2), and Configuration Interaction (CI) codes.
2. Computational Structural Biology—develop methods for modelling components of genomes and a parallel programming environment for structural biology.

> 3. Mathematical Combustion Modelling (MCM)—develop adaptive parallel algorithms for computational fluid dynamics and apply these methods to key problems in commercial burner design.
> 4. Quantum Chromodynamics Calculations—develop lattice gauge algorithms on massively parallel machines for high energy and particle physics applications
> 5. Oil Reservoir Modelling—construct efficient algorithms for parallel systems to model fluid flow through permeable media for better oil recovery methods from wells.
> 6. The Numerical Tokamak Project—develop and integrate particle and fluid plasma models on MPPs as part of a study of Tokamak fusion reactors.
> 7. Global Climate Modelling (GCM)—develop and implement versions of large-scale atmosphere and ocean general circulation models for MPPs.
> 8. Groundwater Transport and Remediation (GTR)—design and implement a multiphase groundwater transport code with interface tracking, fracture flow, microtransport; and
> 9. First Principles Simulation of Materials Properties—develop scalable parallel algorithms for performing local density approximation simulations of materials to novel properties for the Materials Properties Grand Challenge (MPGC).
>
> At the end of FY 1996, Phase I of the GC program was terminated. An RFP was published for Phase II of the program with project starts early in FY 1997. The intent and grant procedures of Phase II are similar to Phase I but the projects now incorporate a infrastructural component to insure that the projects have the computational resources to complete their proposed goals.
>
> (from http://www.ccic.gov/pubs/imp97/48.html)

This is a list that standard literature on the topic easily extends to include the improvement of services such as airline scheduling, mutual fund management, or video-on-demand; environmental modelling of past and potential pollution in air and ground; car crash simulation; integrated design and manufacturing systems; design of new drugs for the pharmaceutical industry by modelling new compounds; simulation of electromagnetic and network properties of electronic systems, and so forth. These numerical models, simulations and applications require a tremendous amount of computing power which goes well beyond the resources of traditional sequential computing systems and can be provided only by massively parallel computing systems (MPPS), capable to complete more than three teraflops (three trillion floating-point operations per second). As a result, it is easy to predict that the next era of computing is going to be dominated by parallel technology. There remains a major, methodological problem, however, which is widely acknowledged but that only time will solve. For decades, the whole world of computer science and technology has focused on monoprocessor architectures. All that work needs now to be re-considered. There is a great need of scientific and technological research on software, operating systems, programming languages, algorithms, applications, hardware implementations, programming tools and methods and other standards for parallel computing. This probably the greatest of all challenges faced by PPC technology.

It may not be a long time before PPCs shall be as common as PCs, and one may wonder whether by then the philosophy of mind will have been influenced by the new computational model of not-ANN parallel processing as much as it has been by Turing machine, neural network, and the now-fashionable quantum computer models.

9.4. Computational Complexity

When searching our magic castle, in the previous sections, we have noticed that strategies (1)/(4) are still insufficient to solve all theoretically computable problems. To understand why, we need to introduce a few more technical explanations.

A computational problem P has a *descriptive* and a *computational* complexity. The descriptive complexity of P is a precise indication of the "semiotic" costs incurred in the description of P in some logical formalism, in terms of amount of symbolic resources required by the codification of P. The computational complexity of P connotes its degree of difficulty, and it is a precise indication of the costs incurred in the computation of the solution of P. It is calculated in terms of amount of resources employed by an algorithm A—or, equivalently, by a Turing machine TM implementing it—to complete its task and solve the problem, independently of the programming language in which the algorithm is written. Both forms of complexity are mathematically related, but in this context we can concentrate only on computational complexity.

We know already from chapter two that the main resources used by a computer are time and memory space, so when we talk about the computational complexity of P we may either refer to its

- space complexity (SC) = the amount of storage memory required by TM to run A and solve P
- time-complexity (TC) = the amount of time required by TM to run A and solve P.

Given SC and TC, the overall complexity of a problem can be classified according to the rate of growth of the memory space and time required to solve it. Now, time complexity is usually taken to be more important because space is re-usable, so let us limit further our attention to TC only.

TC is calculated in terms of number of time steps that the algorithm takes, as a function of the problem size, and for the worst case input. In chapter two, we saw that the size of the data to be manipulated by an algorithm to solve a problem can be quantified in terms of number of symbols and length of tape used by TM as input. Assuming that n gives the measure of the length of the data input x to be computed by TM to solve P, we can organise computable problems into classes depending on their n. The standard way of describing complexity classes is by classifying formal languages Ls that can represent decision problems and can be recognised (i.e. solved) by a TM, but in this context we may adopt a more informal approach. Let us suppose that A is a specific algorithm for solving a particular class of problems C and that TM_A is a Turing machine implementing A. We know that TM_A must perform a certain number of operations O on its x data to solve $P \in C$ before it halts (if TM_A does not halt then O remains undefined, see the halting problem in chapter two) or, more briefly, that $O(TM_A, x)$. The complexity function $f_A(n)$, for all non-negative integers n, gives now an upper bound on the maximum number of elementary operations that TM_A needs to perform on x of size n to solve P. This gives us

- the complexity function: $f_A(n) = \max (O(TM_A, x) \text{ for } |x| = n)$.

The complexity function indicates the *running time* of the algorithm A, but does it also indicate the time-complexity of the corresponding class of problems C? Not yet, because there can be several known algorithms $\{A_1,\ldots,A_m\}$ and therefore TMs for solving the same class of problems. Since they may have different degrees of efficiency, the time-complexity of C or (for the sake of simplicity) C complexity must be defined as being equivalent to the *lowest* degree of complexity of the best algorithm, among all known algorithms devised to solve C, based on the data input set x that results in the *longest* possible time, that is

- $TC(P) = \min (f_1(n),\ldots, f_m(n))$.

The formula indicates how the number of operations required for the solution of P may vary depending on n according to the most efficient algorithm. For example, if the best algorithm to solve P has complexity $TC = n^2$ this means that, when the size of the data input of P doubles, the solution requires at most four times as many operations. We say *at most* because, as in the magic castle, the computer could find the solution before, hence the function sets only the upper bound, the highest degree of bad luck, as it were, indicating the worst case complexities.

Of course, if we assume that TM_A halts and hence that O does not remain undefined, then $f_A(n)$ is bounded, either polynomially or exponentially. To understand how, let us organise best algorithms and corresponding problems into a hierarchy of classes having increasing degrees of complexity. We can start from

1. *the class of polynomial-time complexity problems* (**P**)

These are problems which require algorithms whose time-complexity increases in constant, linear or at most polynomial relation with the size of the input x. A polynomial is an algebraic expression consisting of a sum of terms each of which is a product of a constant and one or more variables or indeterminates raised to a non-negative integral power. If there is only a single variable y, the general form is given by $a_0y^n + a_1y^{n-1} + a_2y^{n-2} + \ldots + a_{n-1}y + a^n$, for example $2y^4 + 3y^3 + 5y + 7$. When the highest power of y (i.e. the degree *deg* of the polynomial), is 0 we have a constant time-complexity function, when deg = 1 we have a linear function, for deg > 1 (in the given example d = 4) we have a polynomial function. In each case, since the number of elementary operations required by TM to solve P with a data input x of size n—that is, the function ($O(TM_A, x)$ for $|x| = n$)—increases with n no more rapidly than a polynomial in n, the algorithm implemented by TM is said to be executable in polynomial time and the problem itself to have a polynomial complexity. The number of operations is never larger than some constant multiple c of n raised to some fixed power k, that is $O \leq c \times n^k$, so the complexity of problems in the class **P** grows slowly enough to make the corresponding algorithms usually computable by a *deterministic* Turing machine (or perhaps a group of them) efficiently in a reasonable time. As an elementary example, you may think of an algorithm to find the treasure as f (n) = 5R, where 5 is the number of boxes to be searched in each room and R the number of rooms. Increases of R will not cause the complexity of the problem to escalate beyond control. Algorithms for problems in **P** are said to be polynomially bounded and efficient for run-time.

If TM is a nondeterministic Turing machine—that is, it consists of a guessing TM that writes a guessed solution on the basis of some arbitrary internal algorithm, and a deterministic TM that evaluates in polynomial time the guesses by a 1 or a 0, to check if the solution is correct—we can speak of

2. the class of *nondeterministic polynomial-time complexity problems* (**NP**)

This is the class of problems for which we can check solutions in polynomial time by a nondeterministic TM. The number of steps required by TM to *check* a solution of an **NP** problem is still a polynomial function of the size of the input. Without the **NP** class, many algorithms that are statistically tractable would not be considered computable within a reasonable (i.e. polynomial) time on the basis of (1) simply because of the limits affecting the linear process of a deterministic TM.

Among **NP** problems**,** we find two important classes

3. the class of *nondeterministic polynomial-time hard complexity problems* (**NP-hard**)

these are problems for which there is a polynomial time reduction to **NP-complete** problems, and

4. the class of *nondeterministic polynomial-time complete complexity problems* (**NPC** or **NP-complete**)

NP-complete problems are the most difficult problems in **NP**. If a problem is **NP-complete** then it requires an exponential time and is extremely unlikely (though it has not been proved) that there is an efficient algorithm to solve it in polynomial time. Two famous NP-complete problems are the

- Travelling Salesman Problem

a salesman wishes to perform a tour of a given number of cities, calling at each city once and only once and travelling the minimum total distance possible, and

- William Hamilton's problem in graph theory

given a n-vertices graph—a collection of vertices and edges G (V,E)—we wish to discover whether there is a Hamiltonian circuit in G (the calculation of the presence of a cycle of edges in G, i.e. a linear path, through which every $n \in V$ is visited once and only once).
In terms of computational complexity, both problems are demonstrably equivalent.

A problem P_1 is said to be a member of the **NPC** class if and only if any other problem P_x in **NP** can be functionally reduced to it in polynomial time (P_x < =*p* P_1). Of course, if a member of **NPC** could be proved to be functionally reducible to a problem in **P** this would prove the more general result **NP** = **P**. Whether this equation is correct represents one the most important open problem in Computational Theory. So far, it is known that **P Í NP** but all attempts to prove that **NP** = **P** have been unsuccessful. This is often, though not necessarily, interpreted as evidence that no **NP**-*complete* problem is actually **P**-solvable, so that **P** ≠ **NP** and **NP** ⊃ **P**. This remains only a reasonable conjecture.

Problems in **P** or **NP** are sufficiently economical in terms of resources required for their solutions. This is no longer the case for **NPC** problems. More generally, computable problems for which no polynomially bounded algorithm is known, such as the **NPC** problems, but can be solved by exponentially bounded algorithms, belong to a higher order class of *intractable* problems, i.e.

5. the class of *exponential* time algorithms **EXP**, whose complexity increases according to a function that raises x to the power of its argument, i.e. $f\ n = x^n$.

Suppose one needs to make 5^R operations to discover the treasure, where R is still the number of rooms in the castle. The algorithm will be $O(5^R)$, and for any extra room Morgan Le Fay adds to the castle to hide her treasure one will have to carry on an exponentially higher number of operations.

Is there any way we may solve problems of class **EXP** via strategies (1) and (3) (running faster and scaling down the space)? We have seen that the answer is in the negative. Now that we know what a **NPC** or **EXP** problem is, we can describe intractable problems as problems which have no polynomial time solutions. There are *completely intractable* problems, i.e. problems that cannot be solved by any algorithm, such as the halting problem, Hilbert's tenth problem (solvability in integers of an arbitrary polynomial equation P = 0 with integral coefficients) or the several problems of "tiling the plane". In chapter two, we have defined them as *undecidable*. Then there are *decidable* but *intractable* problems, which are solvable in principle, but cannot be solved in polynomial time even if we use a nondeterministic system with the ability to pursue an unbounded number of independent computational sequences in parallel. We define these problems as *non-deterministically intractable*. Contrary to polynomial-time problems, exponential-time problems are non-deterministically intractable and their solutions become very quickly impractical even for reasonably small values. One of the most significant examples is the problem posed by the prime factorisation of integers. In chapter two, we encountered the multiplication of two primes 149×193 and the factorisation of their product 28757. We can now say that multiplication belongs to **P**, while factorisation belongs to **EXP**. A five digits number may require an hour work to be factorised by trial and error, and although there are some good algorithms for relatively small integers, less than 120 decimal digit long, a 240-digit integer would already require billions years to be factorised by the fastest algorithms available. This owning to the fact that all universal algorithms devised so far for a "Newtonian" system have exponential complexity. The amount of time they need in order to find the prime factors of a number grows as an exponential function of the size of the number and very quickly the problem becomes computationally intractable, not in principle (recall the unbounded resources of a TM), but in practice. This is why systems for data encryption rely on factoring algorithms. No matter how far we improve the speed and the architecture of our present hardware, there is no way a *classical* computer can discover, in a reasonably short time, the factors that break the code if the number in question is a few hundreds digits long. The potential growth of the computational performances of our present hardware is constrained by the laws of physics. Time is a finite resource, and computable problems which demand too much of it are bound to remain forever unprocessable by monoprocessor machines, even if we know how to reach a solution, simply because there is not enough time to compute it. You may think of such problems as very far planets that we know how to locate but shall never be able to visit, even if a whole colony of human beings was willing to travel for thousands of years.

The treasure in the castle can be beyond your reach. Or so it seems. For, on the one had, we may hope that strategies (2) (improving the algorithms being used) and (4) (parallel computing) will allow us partly to overcome some of our hardware limits. After all, nobody has proved that there can be no polynomial-time algorithm for the factoring problem. On the other hand, there is one more radical strategy, concerning (1) and (3), which still needs to be taken into account. Conditions (1)/(4) try to improve the time-consuming nature of our search. In exploring the castle, we rely on the same basic sequential logic that determines the physical constraint of classic computers. The key-word in classical computing is therefore *sequential time*, no matter whether it is shortened by spreading it over a number of parallel processors. Computational problems with an exponential-time degree of complexity prove that there is a physical limit to how far the *chronological features* of the hardware of a Turing machine working sequentially can be improved. But if *hardware* results constrained by *physics*, we may need to change the former (parallel computing) or the latter. Imagine you could step out of the castle and have a look at it as if you had a plan in front of you. You could see immediately where the treasure is, and get it at once, no matter how many rooms are there. Quantum computing attempts to step outside the physical limits of our standard computers and build a machine which may enjoy such an external perspective, i.e. a discrete, non-deterministic Turing machine in which the physico-topological features of the hardware have been completely transformed. In theory, this means trying to overcome the constraints limiting our present Newtonian systems to take advantage of features offered by quantum systems. Stepping outside is a space move and the new key-word becomes *quantum states*. More in details, this requires a radically new way of codifying data and a new way of processing them.

9.5. Quantum computing

Conceptually, the architectures of a conventional and a quantum computer do not really differ: QCs are Quantum TMs that implement algorithms that obey the laws of quantum physics. Data are uploaded as registers, which are then processed via logic-gates operations to achieve the desired computation under the control of an algorithm, and the results of the computation are encoded as further data registers until the output is finally made available to the user. The essential difference is precisely in the physics of the whole process. A conventional computer encodes, stores and processes discrete n-data registers according to a Newtonian-like physics which allows each bit of data to be only in a fully determined, definite state at a time. Hence a conventional bit is either 0/off or 1/on and can represent only a single value. A quantum computer QC, on the other hand, exploits the possibility of quantum states of atomic particles to store data registers in a *definable* but still *undetermined* quantum *superposition* of two states at the same time. The two logical states (known as "eigenstates") are represented by |1> and |0>, and a quantum bit (*qubit*) is a two-level system that can be set in any superposition $|\mathbf{y}>$ of these two *coexisting* states:

- $|\mathbf{y}> = c_0|0> + c_1|1>$

Now the coefficients c_i can vary, but we may simplify here by adopting the formula a|0> + b|1>, where the coefficients a and b (represented by complex numbers) are the amplitudes of each state normalised constants (so that a + b = 1), and can roughly be thought of as indicating the probability that the qubit will be found in that state (in other words, you can read a|0> as roughly meaning "there is an "a" number of chances that the qubit will be found in the 0 state"). This does not mean that a qubit has only a statistical existence, as some intermediate value between 0 and 1, or that it can be fuzzified. A qubit is actually in both the 0-state and the 1-state *simultaneously*, although possibly to different extents, which can be viewed as probabilities quantified by the amplitudes. It is a vacillating monad of information, and it is only once its state is observed or measured that it invariably collapses to either 0 or 1. This natural phenomenon of superposition, physically present in quantum-mechanical two-level systems such as spin-1/2 particles or two level atoms, is strongly counter-intuitive, for common sense find it difficult to grasp how a qubit could be in two opposite states simultaneously. Unfortunately, conceptual explanations often prove even more incredible and metaphysically obscure than the factual explanandum they try to rationalise or make sense of. In this context, the philosophy student may perhaps take advantage of her acquaintance with Hegel's dialectics and look at quantum superposition $|\mathbf{y}>$ in terms of *Aufhebung* (sublation), a process whereby contradictory positions are reconciled in a higher unity by being both annulled and preserved (aufgehoben) in their synthesis at the same time (an Hegelian example is finite and infinite in the synthesis of the absolute). Once qubits are introduced as "aufgehoben" data units, the next step is to consider quantum logic gates that may perform Boolean-like operations upon quantum-mechanical superpositions of states, to generate other superpositions and eventually the desired output. Special logic gates have to be devised to control the interaction between qubits and generate coherent change in their states. Having modified the very physics of information storage, the implementation of a probabilistic logic of computation of amplitudes must also be adopted. Quantum gates for logical operations such as NOT and XOR on qubits are perfectly possible mathematically. The desired output of quantum computations is obtained through the measurement of the quantum mechanical probability amplitudes of the resulting qubits. Superposition, atomic interference to modify the state of qubits, quantum gates and operations on amplitudes finally make possible a QC whose computations involve two or more different qubits. A QC can now be described as an exponential, non-deterministic Turing machine. To understand its extraordinary computational power, let us compare two data-registers machines searching for a particular pattern among all possible combinations of 8 bits. A conventional 8-bit register computer can only operate sequentially on one of its 256 possible states at a time (2^8, all the 8-digits sequences that can be obtained from combinations of 0s and 1s). To prepare each state of a Newtonian register of n states a classical computer needs n operations. Take now an 8-qubit register QC. Let us assume, for the sake of simplicity, that the amplitudes of the superposition of each qubit are such that measurement of the qubit will result in the |0> state with a "probability" of 0.5, and the |1> state also with a "probability" of 0.5. We can "load" the quantum register in polynomial time to represent all 2^8 states simultaneously, because n elementary operations can generate a state containing 2^n possible states. Each of the 256 combinations has an equal probability of turning up with each measurement, and a single QC can now perform 2^8 operations at once, sifting through all the qubit patterns simultaneously. This is known as *quantum parallelism*, as opposed to standard parallel computing. In theory and simplifying a little, the elementary parallel system introduced in section 9.3 as a network of Von Neumann machines would have to co-ordinate 2^8 PEs to achieve the same result in one operation. Having the whole matrix of 2^8 states in front of itself in one operation, a QC can explore all possible solutions of the problem simultaneously. Constructive and destructive interferences with the qubits suppress irrelevant patterns and enhance those interesting, and the QC finds the particular pattern it is searching for in only one computation. We have stepped outside the castle and found the treasure in a moment, by looking at the map, as it were.

The larger its register, the more exponentially powerful a QC becomes, and a QC with a register of 64 qubits could outsmart any network of supercomputers. Quantum computers, if physically implemented, would then represent utterly new types of machines, qualitatively alternative to our present VNMs. They could perform massively parallel computations, not by organising many processors working in tandem, but thanks to a quantum processor which would operate on coherent superpositions of qubits simultaneously. We can even speculate about the computational power of a network of QCs and quantum super-parallelism. QC could solve, in polynomial time, problems that we have seen require an exponential time to an ordinary computer. This also includes the prime number factorisation, as Peter Shor proved by developing in 1994 the first quantum-mechanical algorithm for factoring efficiently. Thanks to Shor algorithm, a QC with the same clock speed of an ordinary PC could break the keys of any factoring-based cryptosystem, such as the widely used RSA, in a few seconds, while the problem remains simply intractable for the PC itself. Owning to the crucial importance of such a practical application, Shor's algorithm has had a remarkable impact and since its appearance, it has prompted a sudden increase of interest in quantum computing.

Quantum computing is a very young discipline. In 1981, Paul Benioff was among the first to apply quantum theory to computing problems, devising a quantum-mechanical simulation of a Turing machine. The year after, Richard P. Feynman addressed the possibility of a new quantum dynamics for computers that could mimic the behaviour of any finite physical object. In 1985, David Deutsch devised a Turing-like conceptual model of a QC, proving that there is a "universal QC" which can compute anything that a finite machine obeying the laws of quantum computation can compute. Deutsch's important proof provides us with a "quantum" equivalent of Turing Theorem for UTM: there exists a universal quantum Turing machine that can simulate any given quantum Turing machine with only a polynomial slow-down. Since the eighties then, the relationship between information theory and quantum mechanics has become increasingly clearer, together with the potential advantages of a QC and the formidable engineering problems associated with its physical implementation. The potential advantages are intuitive. Such great computational power may force us to re-think the very concept of problems complexity. Not only would QCs make present applications in cryptography, based on factoring difficulties, obsolete, they would also provide new means to generate absolutely secure cryptosystems (based on wavefunction collapse) and, more in general, transform into trivial operations statistical computations that are of extraordinary complexity. Physicists could then use QCs as powerful new modelling tools to investigate quantum physics hypotheses and phenomena that are computationally too complex for our present technology.

Unfortunately, the difficulties in building an actual QC may turn out to be insurmountable. "Newtonian computing" is based on the fact that topological resources (space, location, stability of physical states etc.) are not a major problem, time is. Quantum computing deals with the chronological difficulties of Newtonian computing by means of an ontological shift in the theory of complexity. The relation between computational time and space is inverted (time becomes less problematic than space) if it is possible to transform quantum phenomena of superposition, which are short-lasting and uncontrollable at a microscopic level, into quantum phenomena that are sufficiently long-lasting and controllable at a macroscopic level to enable computational processes to be implemented. Quantum computation is achievable only if this ontological shift becomes empirically feasible, and this in turn is possible only if we can design and build a technology that can

a) lower the technological level at which we can successfully control quantum phenomena. This includes the scalability of technical devices;
b) extend the time-length of the quantum phenomena supporting the computation. This means contrasting the decay process, by which a quantum system dissipates energy in the environment, caused by the coupling of the system with the environment;
c) allow a QC to interact with quantum phenomena indirectly (any direct approach affects the nature of its reference) without causing their decoherence. To perform a successful quantum computation, a QC needs to maintain a coherent unitary evolution of the quantum phenomena involved until the computation is completed. The destruction of the coherent state is known as the phenomenon of decoherence and refers to the high propensity of quantum states to lose their superposition properties—what I have described as their aufgehoben equilibrium—through mutual interaction within the system;
d) satisfy (a)/(c) on a sufficiently large scale to generate a quantitatively usable amount of computational power.

Without (a)/(d) there can be no efficient quantum computations, but it is doubtful whether each requirement can actually be satisfied to a satisfactory degree and, if so, whether the increasing implementations of these requirement are mutually compatible. The difficulty of the overall task of building a QC is shown by the fact that we seem to be running close to wishing a contradiction come true: quantum phenomena should retain their properties and yet modify them. We know that QCs are theoretically possible: the modellisation of virtual quantum machines has determined the development of research during the last decade. But so far the possibility of quantum computing has been proven only mathematically, and in practice no elementary QC with the computational power of a pocket calculator has yet been built or shown to be realistically implementable. Constructing a QC may be as possible as balancing a hair on the head of a needle in a hurricane: nothing contradictory in it, but only God could do it. Down on earth, engineering a real QC means solving enormous difficulties concerning:

- the construction of reliable quantum gates
- the generation of a sufficient amount of qubits
- the "safety" of the qubits (complete isolation of quantum registers from the environment)
- the measurement of the output-qubits.

Wherever a quantum dichotomy/superposition can be generated there lies a potential QC, and several techniques have been devised to create actual qubits, based either on light polarised in two perpendicular directions and hence photons interaction (in theory, photons passing through a stream of caesium atoms could form the basis of a XOR logic gate), on atoms' spins, or more often on cold, trapped ions (atoms with missing electrons, often of Calcium) modified via a pulse of laser light, which can switch the ion from one energy level to the other. Computation could be performed by a series of laser pulses directed at two levels of one ion. In some cases, the technology in question has already been well-developed, e.g. in high-precision spectroscopy, yet the whole process remains so delicate (each pulse would have to operate without exciting any third level, or affecting any of the adjacent ions) and difficult that only a few laboratories in the world have the scientific and technological resources to set up such atomic-scale experiments and, so far, only the creation of a few qubits has proved to be possible (the limit, at the moment of writing, is a ion trap housing at most 6 ions). Unfortunately, a working QC would need thousands of ions, vibrating in synchrony, and even if genetic difficulties could be solved, the construction of a QC may still result an impossible task because of a QC's extreme sensitivity to noise. A quantum system needs to be totally shielded from all interactions with any phenomenon outside itself. Two nearby transistors can switch on and off independently and without any major problem caused by the environment or their mutual influence, but a qubit is a exceedingly fragile creature, whose safety is granted only if absolutely nothing happens to it. It cannot be touched by an air molecule, no light must shine on it, it cannot enter into unpremeditated relations with other qubits, indeed it cannot even be close to other qubits or prematurely measured, for in each of these cases the disturbed qubit stops being in superposition, vacillates no more and settles into a definite digital state, making the whole computation useless. You can then compare a QC to a machine which attempts to tunnel at light speed billions of very fragile crystal spheres (the zeros) and exceedingly unstable detonators (the ones) through extremely bumpy channels. Would the computer be able not to break any one of them? Obviously not, that is why some device to assess the calculation reliability and some process for automatic error correction, similar to those used by ordinary computers, are being investigated also for the construction of a QC. The redundancy approach is roughly that already adopted by Von Neumann himself: each qubit that needs to be "protected" from errors is entangled into a bundle with other qubits whose only task is to work as body-guards, as it were, and make sure the boss-qubit reaches the end of the process safely. To some extent, when an error occurs (decoherence), its presence and impact can be computed and the original piece of information still rescued. To stick to the analogy, a customer buys 12 crystal glasses but you send her 24, just to be sure she is very likely to get at least the right number, if no more. In this case too, some steps have been made towards workable solutions.

Computer science will be increasingly the field where mathematics, nanotechnology, chemistry and physics interact, but whether QCs will ever become a new brand of desktops is hard to say. After all, research into biological computing using large-molecule systems or enzymes as memory registers has already proved to be not impossible, but absolutely impractical, and has been abandoned. For the close future, pessimism about real QC seems to be equally justified: given the exponential timescale of decoherence phenomena, it may not be possible to build a QC beyond the 10-qubit system, but then, who can predict what will happen in a hundred years? New major breakthroughs in the treatment of superpositions, such as present work done on quantum magnetic phenomena, or in the construction of better materials may come to solve the present difficulties. Knowledge of quantum physical phenomena is still limited for utter pessimism to be fully justified. Liquid QC is a good example. The most promising results (a QC that can add 1 + 1, literally) have been obtained by abandoning the assumption that the quantum medium has to be tiny and isolated from its surroundings and using instead "seas of molecules" (the famous cup of coffee) to store the information. In a magnetic field, each nucleus within a molecule spins in a certain direction, which can be used to describe its state (e.g. spinning upwards = 1 and spinning downwards = 0) and Nuclear

Magnetic Resonance (NMR) can then be used to detect these spin states, while bursts of specific radio waves can flip the nuclei between spinning states.

Being equally in doubt, in 1997/8 DARPA funded a 5 million dollar Quantum Information and Computing Institute for investigating quantum computing and its applications. $s may not be sufficient to make something real, but have the extraordinary power of making one feel it possible. All that can be said with some confidence, at the present time, is that work on QCs has been already fruitful for both disciplines involved, so the funds may not be wasted: computer scientists working on the physical constraints of conventional machines have provided new insights into innovative forms of calculation, error correction and data compression, while physicists have been able to investigate powerful ways of engineering and controlling quantum effects. The by-products and spin-offs of quantum computing may well result more important and useful than the project of research that is generating them, at least in the close future.

The usual question now, before moving to the next section: are we quantum computers? The mimetic temptation is constantly lurking and has not spared some philosophers and scientists working in the field. The technology and the physics change, but the conceptual attitude unfortunately does not, nor do the problems affecting it. In a very vague sense, clearly we are "quantum beings". Since quantum physics appears to govern all physical systems at the microlevel, it must also be supposed to regulate our lives as physical bodies in the world. In the same vague sense, then, we are also "Newtonian beings" at the macrolevel, mostly made of water (H_2O makes up about 70% of the human body). Yet, none of us would try to explain the physical mechanics of Brutus' arm stabbing the body of Caesar by working on the level of interaction of masses of H_2O. It seems equally odd that we should try to explain the motivations of the former and the ambitions of the latter by looking at sub-atomic levels of energy superpositions. A perfectly satisfactory description of a computer can be given in terms of a universal Turing machine following algorithms to manage strings of binary symbols, but in the case of an individual, a reliable description can only be given in terms of conscious being, not of biological machine. Of course, the organisational level matters, but it also matters whether it is more important the component or the composite result. Quantum computing has been allied both to computational and non-computational view of the brain, yet analysis of consciousness according to one or the other model seems to follow a neo-Galenic approach[1] whose possible advantage, on the connectionist approach, is that of advancing a step further into the analysis of the atomic components making up a brain. No matter whether quantum neurophysiology is correct, a philosophy of mind based on it seems to be a further step in the wrong, top-down direction towards the explanation of human intelligence. Looking at a single platoon was insufficient to explain the behaviour of the whole army. Checking now the deeds of a single soldier will only make things worst.

9.6. Expert Systems, Knowledge Engineering and Formal Ontologies

[1] According to Galen, the great Greek physician, the human body was composed of four kinds of fluid: phlegm, blood, choler (yellow bile) and melancholy (black bile). Physical and mental characteristics were explained by different proportions of humours in individuals. An excess of phlegm gave rise to a "phlegmatic", or calm, temperament and was associated with water; too much blood caused a "sanguine", or passionate temperament and was associated with air; excess of yellow bile was behind a "choleric", or irascible temperament and was associated with fire; and of black bile caused a "melancholy", or depressive, one, associated with earth. An imbalance of the humours could supposedly be treated by a corresponding diet.

An expert system (ES) is a software application that can solve, or give advise on, a well-defined domain of problems in a particular field, thus performing problem-solving tasks that otherwise may either be impossible or, more often, would require the intervention of a human expert. To achieve such a seemingly intelligent behaviour as well as, if not better than a human expert, an ES usually relies on

1. a knowledge base (KB), i.e. a stored collection of topic-oriented knowledge appropriate to perform problem-solving tasks in a particular field, such as medical diagnosis, insurance planning or investment analysis. The knowledge base of an ES may represent the rules and experience used by an expert to deal with problems in a particular field
2. an inference engine, which generates an output of solutions/advises/answers/recommendations integrating and interpreting inputs represented by (1) and by further data provided by the end user.
3. a interface for the I/O procedures required by (2).

For an ES "knowledge" is a set of facts, linguistically equivalent to declarative sentences assumed to be true, together with a set of inferential rules capable to generate new facts from existing ones. The KB incorporates both relevant data (knowledge-that) and experiences (know-how) that have to be gained from experts and made fully explicit. Once collected, both types of information are converted into the format and sequence needed by the computer to answer the question or solve the problem. The inference engine contains a set of algorithms, known as universal production rules, which embody the general problem-solving methods and heuristic strategies determining the actions to be performed in each case. They have the logical form of an inference: *condition* $\Rightarrow$ *action*, where the "condition" contains a series of patterns specifying the data that cause the rule to be applicable, and the "action" is a set of operations to be performed when the rule results applicable. These general inference rules should not be confused with the topic-oriented rules that are stored in the application domain, as instances of know-how. The inference engine automatically matches data against condition-patterns, and establishes which of them are satisfied and hence which specific rules are applicable. The proper actions are then performed sequentially, so that the ES may add or remove data during the process, thus further affecting the number of rules applicable. The process continues until there are no more conditions satisfied and rules to be applied and the output-solution can be issued. The interaction between end user and ES is normally via question-answer scenarios, and this explains why a friendly interface is important. By entering new data, or simply selecting pre-established alternatives from a list, the end user provides the problem-input. The program normally keeps on asking questions until it has reached a conclusion, which may then be a single suggestion or a list of alternatives, possibly arranged in order of likelihood.

Although no taxonomy is yet widely accepted, through the typology of end-users and knowledge bases, ESs can be distinguished into:

- AES (aid ES), which are meant to help experts to reach a solution. These are often "goal driven": given a goal, e.g. a hypothesis to be falsified, the ES process the available data to reach it.
- RES (replacement ES), which are supposed to interact directly with the layman, replacing the human expert. These are often "event driven": the ES is activated and reacts to data provided by the user, by answering queries and solving problems.
- dynamic vs. static ES, depending on whether the KB remains unchanged during the whole course of the problem solving session, or varies in both depth and breadth in connection with the process
- KBS (knowledge-based system). This is the case when the ES contains mainly instances of

knowledge-that, for example all reference materials available in a field, rather than know-how acquired by interviewing experts. A KBS is a program for relevant data mining and database queries that can also be used to extend the knowledge base itself of an ES.

There are many techniques, especially in the business sector, that may often be included in an ES:

- CBR (Case Based Reasoning) systems, a problem solving technique that can compare and match similar patters of information, representing previous and current situations or cases, and suggest courses of actions based on previously solved occurrences. It is useful where heuristic knowledge is not available, and it is also the natural way for an expert to describe his or her knowledge through examples, stories or cases.
- CP (Constrained Programming) systems that are used to solve complex problems, which contain very large number of variables that can result in a combinatorial explosion when conventional programming techniques are used, for the most efficient allocation of resources. They use specialised "pruning" techniques that can constrain the problem space to a manageable extension. A CP does not always reach the optimum solution, but the advantage is that the proposed solution is usually effective arrived at in a fraction of the time required by an exhaustive, conventional search.

The "intelligent behaviour" of a ES depends as much on the quality of the data stored in its KB as on the logic, encoded in its inferential engine, that controls the application of the relevant rules. Expert systems are often described as being roughly equivalent to human experts, but in practice an AES may perform well above and a RES well below the level of an individual expert. This is why the former may include numeric simulation models, while the latter should be as user-friendly as possible and designed to interact with potential end users who posses only ordinary mathematical and computing skills.

The advantages of ESs are many:

- they can help to analyse and hence acquire a better grasping of human expert knowledge
- they can help to individuate, store, maintain, preserve, disseminate and reproduce expert knowledge
- their knowledge bases can be progressively increased, corrected and updated, although sometimes this implies having to resort to computer programming
- their logic is readily available: ES can explain why further data are needed, how and why certain conclusions were reached, they can describe the reasoning process through back-traces and handle levels of confidence and uncertainty. This is extremely important for the assessment of the results of a session, for it helps to control the value and correctness of the system's conclusions
- they have uniform behaviours: comparably similar problems will always receive comparably similar solutions. This means that an ES may make the same mistakes indefinitely, but will have no psychological or other human bias.
- depending on the complexity of the software and the size of the KB, their implementation may require powerful yet still conventional hardware.
- they are easy to develop using specialised software tools, called ES shells. Shells contain an inference mechanism (backward chaining, forward chaining, or both), they require data to be entered according to a specified format, and may include a number of other facilities, such as tools for writing hypertext, for devising friendly user interfaces, for interfacing with external programs and databases and so forth.

However, some of the short-comings of ESs are equally significant, and should make clear why, again, commercial and scientific implementations of ES are the result of a LAI methodology:

- ES are not universal but have only a narrow domain of "competence" and, as in the case of ANN, their flexibility is a type, not a token property
- they can be brittle, i.e. unable to cope with problems that stretch their rules
- they are non-creative and can hardly learn from past experiences and mistakes with sufficient flexibility
- they cannot resort to common sense knowledge (see the examples provided under the Cycorp entry in the webliography)
- may easily incorporate erroneous rules or badly defined knowledge.

Failure to acknowledge such limits has helped to spread a number of mistaken ideas about ES, three of which are of philosophical interest and should be corrected as follows.

1) Only some kinds of problems are amenable to ES treatment.

ES literature often expresses contempt for the poor expert. She is usually unable to define her skills, experiences and knowledge in such a way as to make them subject to formal analysis by the knowledge engineer and translatable into a logical structure suitable to digital elaboration. Yet, the truth is that ESs can handle successfully only a limited range of well-defined problems or set of initial factors, that can be symbolically represented and whose solutions imply a selection from a definable group of choices and decisions based on logical steps. Broadly speaking, problems that concern not only information management but involve, for example, a reasonable degree of understanding of the issues at stake, are not amenable to ES solution.

2) ES techniques do not resemble human logic in their implementation.

ES strategies may, but do not have to, resemble what expert practitioners actually do. Problem-solving competence qualifies as know-how, hence as practical knowledge. However, practical knowledge is not readily exportable and importable, for it varies deeply, depending on the nature of the subject performing the action. Your knowledge-how to play tennis is utterly different from mine and above all from the knowledge-how that a robot would need to achieve the same result. Hence, to cope with a domain of problems ES techniques need not be like, and actually often differ from what an expert would do in the same situation, although the aim is to reach at least the same result, if not a better one. It is interesting to note that knowledge engineers, being aware of the problem, have devised what is known in the literature as "the telephone text" to determine if a problem is amenable to an ES solution. If expert and end-user can solve a problem via a telephone exchange, then an ES program is probably feasible. The mental experiment is clearly a version of Turing's test. As a positive method, however, it is utterly unreliable. A politician and an economist may solve a number of problems over the telephone, or indeed by email, simply because they rely on a vast amount of commonly shared, yet implicit knowledge, that may not be suitable to sufficiently explicit formalisation. The test is effective only as a negative method: if expert and end-user *cannot* solve a problem by a telephone exchange then an ES program is very likely to be unfeasible as well.

3) ES and fuzzy logic: confusion again.

The reader may have noticed the strong similarity between an ES and a fuzzy system, and indeed many ESs today are based on fuzzy logic. Fuzzy rule-based ESs can cope with degrees of uncertainty and satisfaction of the stated conditions through several methods, including certainty factors associated to rules and user data input, probabilistic inferences and most significant results of tests; yet again, utterly ambiguous, vague or confused knowledge remains useless (recall the GIGO rule). This is why the logical analysis preceding the implementation of an ES is the most difficult stage. In this case too, the myth of a system capable to cope with severe lack of information is to be abandoned.

Knowledge engineering is the branch of computer science that designs ES, translating what needs to be done by a virtual expert system into what can effectively be achieved by a real program. If we are able to do some exercise in problem-solving procedures when discussing parallel computing, in knowledge engineering we can learn how to describe application domains by means of what are technically called, only a bit misleadingly from a philosopher's perspective, *formal ontologies*.

According to a widely accepted approach, the development of an ES includes 5 highly interrelated and interdependent stages. If we concentrate only on the methodological and conceptual aspects, these can be roughly summarised thus:

1. *Identification*, usually in collaboration with domain experts, of all the specific problems (the problem space) to be dealt with by the ES and hence of the kind of KB and inference engine required.
2. *Conceptualisation* of the problem space (knowledge base design). This implies determining the whole set of concepts and objects assumed to exist in the domain identified in (1), their relevant properties, the relations holding among them, the processes in which they may be involved and hence the corresponding control mechanisms of the actions and interactions underlying the problem-solving tasks. The conceptualisation represents the initial stage in the construction of a formal ontology.
3. *Formalisation* of (2) into a logic program (knowledge acquisition and knowledge base construction), which transforms knowledge structures, problems, inference rules, control strategies and information flow into formal representations. The conceptualisation is specified into a formal ontology (see below) in this stage.
4. *Implementation* of (3) into the language of the development shell and realisation of a working prototype.

An ES can be implemented using standard programming language like C or Visual Basic; languages specifically designed for AI applications, like LISP, which we have seen provides special tools for manipulating lists of objects that have been found to be an underlying aspect of human reasoning, or PROLOG, which provides an inference engine; or by using expert system shell. A shell not only comes with an inference engine, so that the knowledge engineer only needs to supply the relevant knowledge, but it also provides reasoning tools such as

- rules for inferring new facts
- demons which constantly check the current status of the ES, and activate if a set of conditions are met
- when-needed methods. If the ES needs some data, a when-needed method for those data activates to obtain it from the user of the ES
- when-changed methods. If key data in the ES change, this method activates and performs necessary actions relevant to that change.

5. *Testing* of (4), including verification of accuracy.

Epistemologically, the most interesting stages are the second and the third, and two points are worth remarking upon, before we address the ontology issue. Firstly, working ESs are sometimes constructed by starting from what they can really do, that is the problems they can successfully handle, rather than the needs of the end users. This is the result of a clear understanding of the distinction between problems that are amenable to ES treatment and problems that result untreatable, but it also clarifies why the end users may often be dissatisfied. It is the task of the knowledge engineer to keep the right balance between ES capacities and users' expectations. Secondly, as far as the intelligent behaviour of an ES is concerned, clearly the design and construction of the KB and the inference engine exploit the experts' know-how and the knowledge engineer's capacities. An ES should then be understood as a sort of interactive archive of procedural and factual knowledge and not been confused with GOFAI projects.

Looking now at the construction of an ES from the perspective of the design of the knowledge base rather than of the problem-solving tasks, stages (1)/(3) can be made more specific by comparing them to the three stages through which knowledge representation (KR) in a declarative language is achieved:

1. selection of a domain of discourse or application domain, which can be referred to by using the concept of *possible world* (PW). By a PW I mean here a consistent scenario constituted by a set of entities with their properties and, possibly, their inter-relations and processes. It may but usually does not have to be already instantiated
2. conceptualisation (that is, an abstract, informal and simplified view) of the PW we wish to represent formally. A conceptualisation encompasses a description of the concepts appropriate to, and the entities assumed to exist in PW, together with their properties, (possibly) the processes to which they are subject and, above all, the relations holding among them. The set of entities populating the PW is called the universe of discourse (note that this is often taken to be the truly first stage of KR).
3. a specification of the conceptualisation of the PW by means of a formal ontology Ω, which explicitly regiments the abstract conceptualisation into a logical model. The model may then become a concrete data structure amenable to processing. A formal ontology provides

- a vocabulary, i.e. a mapping from the names of the entities in the universe of discourse to representational terms (the names of the defined entities, for example classes, instances, n-ary relations, functions, and individual constants). The terms thus represent the subject-oriented knowledge representation primitives of the ontology
- a semantics, i.e. a set of definitions of the univocal meanings of the representational terms (the properties of the entities), based on other terms (simple equations), on conservative introductions (non-informative definitions of semantically empty labels) or "contentful" axiomatic postulates.
- a syntax, i.e. a set of axioms describing the actual or possible functions and mutual relations among the terms, that fully regulate their correct uses.

More intuitively, formal ontologies provide a vocabulary of formal concepts with a description of the mutual relations and of the ways in which they can be correctly used to represent, and (possibly) share, domain-oriented knowledge bases. An ontology Ω, together with a predicate logic establishing the vocabulary of logical symbols, their syntax and semantics, may then be understood as representing the language of KR and the basis for the construction of an expert system.

Ontologies were initially developed only as part of KB designs, but they clearly have enormous potentialities as mapping tools for repositories of organised knowledge in any context where it is important to have a full and explicit documentation of the domain of discourse, as Intel's WAVE project demonstrates (see the webliography). This pragmatic perspective must be kept in mind to understand three exigencies. Capturing and representing domain-oriented knowledge are two essential stages in the construction of any ES. Unfortunately, both processes are difficult, time consuming and therefore expensive. Consequently, while many shells are available for developing production rules systems, realising the ontology from scratch remains one of the major costs in building an ES. The solution is to design ontologies that may be both *upgradable* and *reusable*, so that it becomes possible to maintain, increase, correct and improve them easily through time and import/export them across different projects. For example, the legal ontology of an ES dealing with human rights could be imported within the medical ontology of an ES dealing with patients' rights in a hospital and new concepts, developed within the medical ontology, could be reused to upgrade the legal ontology. Knowledge sharing is of utmost importance for any collaborative project but cannot be based only on reusability: inter-operability, based on interactive communication by way of message-passing, is equally vital. Special-purpose ESs can communicate with each other only if they can commit themselves to the same ontologies. A *sharable* ontology is therefore designed to allow such a degree of commitment, on the side of intelligent agents (groups of people and/or programs, including ES and other KB systems), as to make their efficient interoperability unproblematic. An ontology that allows future development (upgradable), inclusion into another ontology (reusable) and communication with other ontologies (sharable) is a *open ontology*, which minimises knowledge waste and redundancy, enhance collaboration and can be used both as an open catalogue of the subject domain and as an open vocabulary through which co-operating agents can interact. An ontology that satisfies all these conditions can be defined as *canonical*.

It was only in the nineties that "green" methodologies that minimise waste and maximise reusability were adopted to promote standard KR languages and protocols for the construction of canonical ontologies. In this, a fundamental role has been played by the analysis of the concept of ontological commitment.

An ontological commitment is the procedural commitment *to* an ontology Ω and must be distinguished from a commitment *of* an ontology Ω to a particular methodology. Literature on the topic often oversees this distinction, so in this context I shall restrict the use of the technical expression to the former case only, and speak of Ω-commitments in the latter. We have seen that an ontological commitment consists in the explicit acceptance of an ontology among a community of intelligent agents. Now, recognition of degrees of commitments may be a purely "behaviourist" matter, for it refers only to the description level of the "epistemic" actions of an agent (knowledge-level) and not to the description level of its internal structure (the symbol-level representation used in the agent's internal knowledge base). According to such a behaviourist approach, two agents are said to be committed to the same ontology if and only if their observable actions are consistent with the definitions provided in the ontology. Although a plurality of heterogeneous agents may not refer to exactly the same ontology Ω, if they are committed to one they are capable to use a minimal vocabulary to pose logical queries and make logical assertions that are consistent with the description of the PW in question. If the actions of two agents are only Ω-consistent and the two agents cannot yet interoperate through Ω successfully because they commit themselves to completely different areas of Ω, then the two agents are said to be only minimally committed to Ω. If they can also interoperate by exchanging queries and assertions, then they are said to be also Ω-interoperative and maximally committed to Ω. The case of two or more agents sharing exactly the same ontology Ω is then only a specific case of maximal commitment, which grants that their inter-operability is *complete*.

Ω-commitments have attracted less attention than ontological commitments but are of comparable importance. They can be organised into five types, and every intelligent agent shows a particular attitude about each of them, either explicitly or implicitly.

a) *particularists vs. universalists*. Libraries of specific ontologies are usually constructed bottom up, by looking at particular domains of discourse. For example, the library of Ontolingua (see the webliography) contains ontologies such as 3D-Tensor-Quantities, Abstract-Algebra, Basic-Matrix-Algebra, Bibliographic-Data, Chemical-Crystals, Chemical-Elements.

On the other hand, it is also possible to construct general ontologies top down. This was the project carried on by the Cyc (see webliography), which attempted to construct an ontology of common-sense knowledge about the world that could then lead to more specific local ontologies. From the perspective of LAI, a "particularist" approach is to be preferred, for it grants almost immediate applicability and, when developed by a "green" methodology, ensures the modularity of the achievements. However, since specific levels of abstraction may vary depending on what the ontology is used for, and since, in terms of inclusion into another ontology, the more abstract an ontology is the more likely it will be reusable by other projects, it may be worth attempting to develop complete and coherent *core ontologies* (see Valente and Breuker's online article in the webliography) by means of a combination of top-down and bottom-up approaches, working at levels which are higher than the "particularist" yet lower than the "universalist", as in the case of canonical ontologies concerning whole fields such as medicine or computer engineering domains.

b) *Platonist constructionism vs. nominalist constructivism*. After (1), the next commitment is to the specific PW that needs to be conceptualised and "canonically ontologised". Note that here an a priori nominalist commitment regarding the existence of PW and its entities is probably harmful. Although physically instantiated PWs may presumably receive particular attention, a more tolerant, Platonist approach seems preferable, and knowledge engineering, being driven by real-world demands, has already included the ontologisation of non-physical domains as well, as the few examples from Ontolingua mentioned above already suffice to prove. In a Platonist ontology with a *constructionist* approach, existence is equated to static representability, hence to logical consistency, both internal (stand-alone) and with respect to the system (networked), so the *construction* of a possible entity as a set of properties which are not contradictory in either respects is perfectly acceptable and constitutes the fundamental criterion for ontological admission. Philosophically, it is the objective consistency of a set of entities that makes possible the performance of subsequent actions. In a nominalist ontology with a *constructivist* approach, on the contrary, existence is equated either to actual existence, physically experienceable by a subject, or to algorithmic feasibility (perhaps via the analysis of a dedicated Turing machine) and dynamic provability again by a subject, so "merely" non-contradictory entities do not yet qualify for an ontological treatment. Philosophically, it is the range of subjective doings that settles the limits of what may count as being. However, since for an intelligent agent its own objective (agent-independent) ontology, with its description of the entities and their mutual relations, is all there is and represents the condition of possibility of any further actions, and since it is at least immaterial whether the objects in question are postulated as possible, identified as real, or accepted as finitely constructible, a Platonist-constructionist approach is not only viable but also preferable, for it allows a wider class of ontological libraries. In other words, ontologies are designed to support practical needs and a constructionist policy allows to satisfy more of them. For scholarly problem-solving tasks, an ontology of Shakespearean characters may well be as acceptable and useful as an ontology of printed editions of Shakespeare's works, while a mathematical ontology will be easier to handle if the law of excluded middle and proofs by means of *reductio ad absurdum* are available.

c) *substance vs. function*. The third commitment is to the type of conceptualisation of the PW to be adopted. In principle, a conceptualisation could also be function-centred but in practice, it is usually object-centred. This is in line with the commitment in favour of a consistency-based ontology, made in (b), and influences both the range of possible ontologisations in (d) and the kind of logic adopted for their design in (e).

d) *static vs. dynamic types of ontologies*. Consistency-based PW and object-centred conceptualisations tend to generate static ontologies, based on structures of entities, class of entities, n-ary relations and functions. Dynamic processes are left implicit, translated into relations or functions, or often disregarded. One may wonder whether things could not be improved.

e) *the many logics of a formal ontology*. What logic is more suitable for the design of an ontology and hence for KR? Ontologies appear to be usually committed to classic first-order predicate logic (a two-values logic including the law of bivalence), but second-order logic and non-standard or deviant logics (such as modal logic, n-values logics and fuzzy logic) may also be taken into consideration in the future, at least for representative needs, if not for inferential requirements. It must be remarked that reluctance towards the use of deviant logics has often stemmed from nominalist attitudes, such as Quine's, which we have already remarked to be unjustified in this context (see (b)).

The previous Ω-commitments can be summarised by saying that the philosophy of canonical ontologies is constructionist and in line with a LAI approach, but that many methodological issues are still wanting further study and clarification. Some of the most interesting problems, either conceptual or methodological, concern

- the logic of formal ontologies, including the study of the relations among sets of ontologies and of the diachronic development of ontologies
- the possibility of a standard, ontological "interlingua" that may work as an ontological foundation of domain-oriented canonical ontologies
- the distinction and investigation of meta-ontologies (ontologies of ontologies) vs. ontological "super theories", i.e. general and abstract ontological tools such as set theory, mereology, topology, graph theory, systems theory, first and second order logic and modal logic
- the exploration of ways in which problems posed by the typically multi-faceted character of ontologies (the presence of several viewpoints on one and the same domain of discourse, think for example of your role as an entity in the ontology of your institution and in the ontology of your cricket team) could be solved
- the relation between formal ontologies, conceptualisations and taxonomic criteria (categorisation) of possible worlds
- the expansion of ontologies as to include causal, procedural and spatio-temporal languages, to mention only a few.

In working on similar problems, epistemology and knowledge engineering can be of mutual assistance. In epistemology, it is to be hoped that, on the one hand, we may be able to exploit the methodology of formal ontologies to study conceptual systems as well as to develop libraries of ontologies of ideas treated as artefacts/entities, and that, on the other hand, we may avail ourselves of the conceptual apparatus of formal ontology to articulate a general theory of critical constructionism, including the literary constructionism outlined in chapter four. In knowledge engineering, epistemological and methodological analyses may provide further conceptual clarifications and explanations to improve the present status of the field and help to design better applications. In 1996, in the presentation of the "Ontological Engineering Workshop: Objectives and Background", it was remarked that

> Ontological engineering currently is a craft rather than a science. [...] In workshops and in the published literature, ontologies are called important or indispensable in designing and building knowledge-based systems and in facilitating sharing and reuse of knowledge resources. It is evident that quite a number of research groups are building ontologies. It is less clear what design decisions are taken and how they contribute to the success (or failure) of the ontologies developed. Thus the danger of making the same mistakes over and over again is as real as the danger of inventing the

> wheel at several places simultaneously. Development of an ontology is a laborious and therefore costly process. The field would profit immensely if we could learn from each other's successes and failures. One way to lay down the do's and don'ts is in the form of systematic design practices. With so many groups working on the subject, it is possible to gather experiences and start articulating systematic design practices. (from http://wwwis.cs.utwente.nl:8080/kbs/EcaiWorkshop/objective.html)

Obviously there is plenty of useful work that philosophically minded engineers and mathematically minded philosophers can do together in this field.

9.7. Robotics Cybernetics and Artificial Agents

CHEAP LABOR. ROSSUM'S ROBOTS. ROBOTS FOR THE TOPICS. 150 DOLLARS EACH. EVERYONE SHOULD BUY HIS OWN ROBOTS. DO YOU WANT TO CHEAPEN YOUR OUTPUT? ORDER ROSSUM'S ROBOTS.

One of the many ads on Internet? Not at all. These famous lines come from Karel Capek's play *Rossum's Universal Robots*, which opened in Prague in January 1921. They contain the first occurrence of "robot", a term coined by Capek and his brother from the Czech word "robota", which literally means "forced work or labor", coming from the Latin "robor" ("power", "force"). The play is staged in a factory that builds artificial agents. The plot centres around the conversion of such workers into military forces that eventually take over the factory and exterminate the whole of humanity. The end leaves open whether the robots, which have lost the necessary know-how to reproduce themselves, may be on their way to become more human.

Philosophically rich and controversial, *R.U.R.* was unanimously acknowledged as a masterpiece since its appearance, and has become a classic of technologically dystopian literature. The neologism "robot" was, among other things, greatly successful. Already in 1924, we find that the Soviet Union propaganda film *Aelita* became also known as *The Revolt of the Robots*. Twenty years later, the young Isaac Asimov, who was born in Russia in 1920 and was well-acquainted with Capek's terminology, introduced the word "robotics" in English, to indicate the scientific study and application of robots (*Runaround*, 1942).

Born as science-fiction terms, in the last thirty years the concepts of "robots" and "robotics" have gradually come to be widely accepted in the scientific community and have merged with *cybernetics*, a term coined by Norbert Wiener from the Greek word "kybernetes" ("steersman" or "governor") to indicate the interdisciplinary science of communication, control and regulatory properties of complex, interacting systems, both biological and mechanical. Cybernetics developed during World War II as the investigation of the techniques and automatic-control mechanisms by which information is transformed into desired performance and then information concerning the actual results of the performance is made available to the system itself to determine its future course of action, through control mechanisms for feedback and self-correction. Theoretically, cybernetics had a much wider scope of scientific interests than robotics, for it could be viewed as the general science of systems of communication and control, both in living organisms and in machines, capable to acquire, process, communicate and employ information to perform practical tasks, but in its technical applications, as a branch of automation engineering, it has gradually become hardly distinguishable from it.

Whether the fruit of a robotic or cybernetic approach, artificial agents are a very old fantasy. The ultimate ideal in the mechanisation of production is the perpetual motion machine: a machine that could produce more useful work or energy than it consumes, or a machine that, once set in motion, would be at least 100% efficient at converting energy into work and hence could continue to be in motion indefinitely, without requiring any further input of energy. The first and second law of thermodynamics have proved both such projects to be impossible. Similarly, the ultimate ideal in the automatisation of action is the intelligent automaton, an artificial agent having its own power source, designed to be able to behave and respond to the environment independently of, and as intelligently as any other ordinary human agent. For some time, computer science has tried to make this hope come true, yet since the eighties there seems to be a tacit acceptance that the GOFAI project has failed too. In this case, however, there are no laws, comparable to those of thermodynamics, stating that forms of non-biological intelligence are impossible, so robots shall always count as possible entities in principle.

Mechanical automata are known to have been engineered as toys, or more often imagined as fictional creatures, for thousands of years and in utterly different cultures. Today, artificial agents can be distinguished into four families:

1) *androids*
2) *cyborgs*
3) *robots*
4) *webbots*

The order in the taxonomy is not merely alphabetic but also conceptual and technological.

The word "androi" in ancient Greek meant "men", and androids (or simply "droids") are biogenetically engineered agents, composed entirely of organic substance. They are mimetic, human-like models, such as Frankenstein's, Rossum's and the famous NEXUS 6 Replicants that are to be "retired" by Deckard/Harrison Ford in *Blade Runner*. In the film, replicants differ from humans only in that they are emotionally immature, cannot help being completely self-controlled or lack any empathy. In one draft of the script we are told that a body is recognised to be that of a replicant only after 2 hours autopsy. As with Frankenstein's and Rossum's creatures, problems arise when the replicants begin to acquire a fully-human nature. For example, the new prototype Rachael is endowed with experimental memory that provides her with a past as a background for her emotions. She takes for granted the fact that she is human. Conceptually, androids are the first and most natural kind of artificial agents ever conceived. Technologically, they belong to the realm of science fiction and philosophical mental experiments.

CYBernetic ORGanisms (*Cyborgs*) are partly human and partly mechanical agents. Although Wiener himself did not discuss the actual physical merger of living organisms and machines, their design originate from the field of cybernetics, and their actual possibility was suggested by Manfred Clynes and Nathan Kline, two NASA scientists who coined the word "cyborg" to refer to the potential advantages of altering, replacing or augmenting the body of astronauts with exogenous components to make life in space more comfortable. In a generic way, many people could qualify as "cyborgs", having parts of their body replaced or augmented by prosthetic devices, false teeth, mechanical heart valve, contact lens, artificial bones, or dependent on mechanisms such as pacemakers and automatic biochemical pumps. In a more fictional sense, however, a true cyborg is an agent who has had his or her central nervous system mechanically or digitally tampered. In theory, we could imagine having memory chips implanted in our brain and directly consultable "from within". Anyone could become a walking encyclopaedia and never forget dates or telephone numbers. More seriously, research in artificial ears and eyes—miniature video camera wired to an electrode array implanted in the visual cortex of certain blind people—has been in progress for some time. Of course, the immense technological problems concern the way in which the biological neural network and the various mechanisms in question may be interfaced. So far cyborgs have remained either poor human beings with a slightly improved body or fictional characters.

With the robots we enter in the realm of the technologically feasible. *R.U.R.*'s robots are, in our terminology, androids, but the tension between biochemical reproduction and mechanical invention is already present even in the play. The old Rossum is a scientist who wishes to prove that a human being can be constructed in a laboratory. Although his techniques are different, his philosophy is as materialist and mechanist as Dr Frankenstein's. Contrary to the latter, in the play he is said to have failed in all his attempts to create a perfect human being. The young Rossum, his son who overtakes his project, is a pragmatic engineer, whose only aim is to provide cheap labour force. His new robots are still biochemically-based, but lack any human features not directly useful for the performance of their industrial tasks, in short any emotional life. It is tempting to view the two Rossums as personifications of GOFAI and LAI respectively; and in so far as the former holds a mimetic and materialist perspective, while the latter limits himself to a task-oriented approach, the comparison may partially hold. Its obvious limit, of course, lies in the overall idea of a carbon-based form of artificial and intelligent life. Real robots represent the merger of mechanical automata and computerised control systems. They do not take on a rather useless human appearance, such as *Star Trek* Data, and at most they may vaguely resemble a limb, as in the case of industrial arms.

According to a fairly broad definition, a robot is any automatic machine that can be programmed to perform a variety of manual tasks, involving moving and handling objects, which would require some intelligence when performed by a human being. The problem with this definition is that it would fit even a washing machine: washing one's socks requires some intelligence, and the task is successfully achieved by the machine in its own way. Only an artificial agent would wash a pair of socks by quickly rotating them one way and another in a lot of hot water and soap for about an hour, but there is nothing intrinsically wrong with the procedure. However, we do not consider washing machines and dish washers authentic robots. Why? Because they create the environment with which they can interact. A washing machine is in fact a whole microworld within which a particular task, and only that task, becomes achievable. It does not actually interact with the same environment in which we find ourselves as bodies, nor is it ready to cope with novelties coming from it. On the contrary, this is what computer-controlled industrial arms try to achieve, at least to some interesting extent.

To interact with its environment (*work envelope*) successfully, an industrial robot needs to be able to reach any possible point P in its space. The basic directions of movement in space are 4: up and down, side to side, forward and backward and rotation, and we say that the arm has 1 degree of freedom for each direction a joint can go. Now, any P can be described by a set of 3 Cartesian coordinates {x, y, z} and an arm can reach any P only if it has at least 6 degrees of freedom. The human arm enjoys 7 of them, for the shoulder has 3 (up-down, side-to-side, rotation) the elbow 1 (up-down) and the wrist another 3 (up-down, side-to-side, rotation), while jointed-arm robots can have from a minimum of 6 (in this case they are like a human arm, except for the rotation of the shoulder) up to 20. The structure of a basic 6-degrees jointed-arm robot consists in:

- a digital controller, i.e. a computer that coordinates the movements of the mechanical arm and may network the robot with other digital devices
- an end effector, not necessarily resembling a human hand, suited for a specific task. It could even be a simple screwdriver and may be replaceable with other tools.
- sensors that can provide some feedback to the robot, keep it informed about the environment's and its own state and thus correct its movements if they deviate from the programmed patterns.

- an arm made of joints and links, whose task is to position the end-effector and sensors according to the instructions provided by the controller
- a drive, that is an engine that drives the links into their desired position.

Similar robots are the outcome of the merger of the industrial revolution and the information age. In 1956, George Devil and Joseph Engelberger, known as the father of robotics, formed “Unimation” the world's first robot company. Their vision was to construct industrial robots that could be integrated into a factory in the same way as business machines were part of an office. Five years later, General Motors adopted the first robot-worker, and in 1997, thanks to the advancement in microelectronics, there were almost half a million robots in the world, a number that was likely to increase. Industrial robots are more efficient than human workers, at least under three aspects: improved management control and productivity, no loss in continuous performance, consistent high quality production, and thus they are more competitive. They can do jobs that would be too dangerous, boring, repetitive or unhealthy for human beings, thus improving the quality of life and of working conditions. In addition, they can help to lower the price of manufactured goods through the implementation of highly automated assembly lines.

Most robots are still only sophisticated variations of the model illustrated above; almost all of them are used in the car industry, in electronics factories, aircraft manufacturing, warehouses and laboratories, but it is likely that in the future a wider variety of robotised agents will become more common in other contexts as well. Along this process, a more consistent adoption of a LAI approach will probably result essential. Our present industrial robots are nothing more than microcomputer-controlled manipulators, entirely pre-programmed. Their mechanical arms are trained by physically moving them through the required motions, so that the controller can record all the movements and repeat them precisely. More generally, all kinds of robots, including mobile ones, lack independent processing of information on a wide scale, they are not creative or innovative, can take at most a very limited number of decisions, do not learn or significantly improve their behaviour through repeated trial and error, and have very little adaptability (as in the case of artificial neural networks, robots’ adaptability is mainly a type, not a token feature). To make things worst, professional robots can cost anything between $50,000 and $200,000. Yet none of the previous factors is sufficient to prove that they may not become as common as refrigerators or cars, artefacts which share with robots all the previous limits if not more (their price is very likely only a consequence rather then a cause of their small diffusion, and may dramatically decrease in time, like that of cars). The problem seems rather one of general strategy. Robots flawlessly perform specific jobs; there are a lot of jobs that we may like them to do for us, from cutting the grass in the garden to cleaning the carpet or driving our car home when we are too tired or have enjoyed an extra glass of wine. The question is that we still need to find a way to adapt the performance of such jobs to the capacity of our present or potential robots. We all wish to play the role of the “Sorceror's Apprentice”, but like Mickey Mouse we have not yet been capable to transform tasks, which would require our intelligence to be performed successfully, into stupid tasks a robot may safely take care of, no matter whether less economically than us (the washing machine example) and perhaps even better than we do. On the one hand, there is a need to rethink the methods whereby something can be done: consider how differently the washing machine operates from a human being. On the other hand, we need to transform the environment in which the task is performed and adapt it to the robots’ capacities. Only when gardens will be shaped and modified so as to make possible for a robot to cut the grass, and streets will be constructed to allow robotised buses to travel fast and safely will the relevant robots become a commodity. It is the environment of a robot that can become a bit more artificial, a contrived microworld in which objects, properties, relations and events are as narrowly and explicitly defined in advance as possible.

Two final considerations may now be in order. The first is about a social issue. Robots are often viewed as a cause of unemployment, yet this is a superficial mistake, for more than one reasons. Firstly, there are less than 500,000 robots currently active in the world and, in many cases, they are doing jobs that human beings should not or could not do anyway. Secondly, robotics has generated quite a number of new professions and jobs in other areas. Finally, in case the problem would not be represented by a robotised, job-less society, but by a society in which job-less people cannot live comfortably. Having a job, as opposed to having a practical activity or interest, is hardly a human need. It is an unfortunate necessity. Ask any student who has not yet become a workaholic. In so far as a robotised society produces more wealth and free time and both resources are fairly distributed among its members, we may all wish to take our holidays and let our mechanical fellows work for us. This is a typical philosophical statement, one may object. The reply is that at best, this is a social and political issue to be solved. At worst, it can only be a sheer utopian fantasy, not a dystopia. The second consideration is ethical. In the philosophy of robotics we do not normally wonder whether we may be mechanical agents, but rather whether robots may ever achieve a human status. Intelligence, consciousness, emotional life, bodily experience, communication skills and social relations are usually, if variously, recognised as the essential elements that would transform a robot into a person, and the possible acquisition of such properties, besides firing some idle speculations, unmasks a more interesting, master-slave dialectic that seems to affect more generally our entire conception of co-operation and interaction with other entities, no matter whether artificial, animal or human. Robots are the equivalent of slaves. It is indicative that in *Star Wars* both R2D2 and C3PO are realistically viewed as friendly servants enjoying no rights. Now, masters wish to have autonomous agents (animals, slaves, robots, employees, assistants etc.) capable to perform intelligent tasks on their behalf and possibly independently of their constant intervention. Intelligent “agenthood” is therefore intimately connected with the notion of delegation, which inevitably brings with itself trust, accountability and responsabilisation. Necessarily then, the more the master delegates the agent, the more the latter needs to become independent and autonomous, the more likely it is that the master may lose control over it. Loss of control may then imply three different risks. On the one hand, the agent may not perform its task satisfactorily or as well as the master would have done in its place. This is not a major problem, as long as the master is able to regain control of the task in question or fix the problems. On the other hand, if the agent performs its tasks perfectly well, the master may become wholly dependent on the agent (think of the boss without his secretary, the robotised factory without its robots, the professor without her teaching assistant) thus reversing the positions; or finally, even without dependency, the agent may, in any case, become completely autonomous and free itself from the master (so far this is possible only among human beings). The master is at least vaguely aware of such a dialectic and, in order not to allow the agent to become completely autonomous of himself, establishes rules that enables him to retain some essential control over the agent’s behaviour and well-being. In robotics, Asimov translated such rules into his famous 4 laws:

0) A robot may not injure humanity, or, through inaction, allow humanity to come to harm.
1) A robot may not injure a human being or, through inaction, allow a human being to come to harm, unless this would violate a higher order law.
2) A robot must obey the orders given it by human beings except where such orders would conflict with a higher order law.
3) A robot must protect its own existence as long as such protection does not conflict with a

higher order law.

The laws are clearly the master's rules and are far from being "robotically correct". Now, in human societies similar laws are not written, but this is precisely the advantage to deal with intelligent agents, they grasp them on the fly when the get their first job, so you do not have to codify explicit instructions to make them understand the rules of the game. A careful study of our behaviour towards robots' actions not only is epistemologically relevant to our understanding of perception and agenthood, it may also result significant to increase our grasping of the rules governing social structures and interactions.

Industrial robots have deeply affected their working environment to make possible their successful interactions. The industrial architecture of robotised factories is very different from the industrial architecture of "human" factories. This is reasonable. The more compatible an agent and its environment become, the more likely it is that the former will be able to perform its tasks efficiently. Let us define as "ontological enveloping" the process of adapting the environment to the agent in order to enhance the latter's capacities of interaction. We have seen that, presumably, home robots will have to prompt a higher degree of ontological enveloping to become everyday tools. Are there robots working in a totally enveloped environment? The answer is in the positive. We call them webbots.

Webbots are a new class of software, as one may easily guess from the origin of the word (WEB+roBOTS, the term is also used by html authoring tools, such as Microsoft FrontPage, to describe applications that allow simple interactive components to be added to Web pages without coding any instructions). Broadly speaking, webbots can be described as body-less, fully-digital robots, situated within a cyberspace (it could be the digital space of your hard disk, of a company's Intranet or the global space of the Internet) that operate interactively on another agent's behalf (usually a single end user, but it could be a laboratory, another webbot, etc.) and can only act on other digital entities in the same space, though they themselves can be operated upon from without, by a human agent, and may "migrate" from one space to another, like any other software. They usually include a NLP (natural language system) interface for access. Webbots can for example search and query their environment, interact with it over time by performing information-based tasks that have been assigned to them by other agents, and thus pursue a large variety of different goals. A webbot may be an IRC or MUD program (a Robo-Sysop) that provides some useful service, such as preventing random users from adopting nicknames already adopted by others, or simply chatting with you when nobody else is online, like Julia (how would you feel if you were to discover that the Internet fascinating patner with "whom" you had a rather gallant conversation was "only" a chatterbot?). Alternatively, it may be a cartoon that speaks to the human user and acts as a guide or teacher, or a Web program (sometimes also known as crawler or spider) that automatically explores the cyberspace to retrieve documents, thus making possible search engines. A webbot may simply be an utility that constantly keeps clean, compressed and virus-free the hard disk of your computer. Webbots can be personalised to satisfy one' requirements, and this is a step forward with respect to the lack of flexibility shown by other intelligent agents. Predictably, the same webbot will in the future be able to perform a wide variety of completely different tasks, instantiating real token-flexibility. Their tasks are often humble, but they are getting increasingly more substantial, and have started being employed in many business applications in the financial, manufacturing, retailing, e-commerce and travel industries. Their future includes didactic and ludic applications. Cyberspaces would be far more chaotic and polluted places without them. Webbots' power lies in the total ontological envelopment of their environment, for they are made of the very same digital stuff cyberspace consists of, and in the uniformity of their tasks, all information-based. Their potentialities have just begun to be exploited, but they already are one of the most successful application of LAI.

10. The Limits of LAI

At this point we should resist the temptation to assign no boundary to what LAI may be able to achieve in the future. The remarkably ingenious efforts made to transform a computer into something smarter than a mere electronic Abacus have always had the great advantage of being cumulative, but must remain subject to the conceptual limits of the technology in question. The "intelligence" of a computer lies all in its capacity to detect and process a *relation of difference* (usually, but not necessarily, binary), and proceed inferentially on its basis. It is an extraordinary capacity in itself, and a very powerful one too, when it is associated with the right logic—one can do a lot of extraordinarily complex mathematics or win a chess game with a Master just by carefully exploiting it—but whenever conditions (a)/(e) (computability, epistemic-independence, experience-independence, body-independence and context-freedom) are substantially unfulfilled, it nevertheless remains completely insufficient to *emulate* even the elementary intelligence of a guide dog or a sheepdog, which grasps and deals primarily with the *relata* themselves. We must not forget that only under specially regimented conditions can a collection of detected relations of difference concerning some empirical aspect of reality replace direct experiential knowledge of that aspect of reality. Computers may never fail to read a bar code correctly, but cannot explain the difference between a painting by Monet and one by Pizarro. More generally, mimetic approaches to AI are not viable because knowledge, experience, bodily involvement and interaction with the context have all a cumulative and irreversible nature. For a human being (but the same may be extended to other animals), to know how to speak a language or demonstrate a theorem is not equivalent to mastering the whole sum of instructions required to perform well the specific task in question, nor does it consist in establishing a one-to-one relation between skilled capacities to do x and the correct list of instructions to do x. Any acquisition of a special skill or experience or capacity by a human being means the loss of a particular virginity. Once we have learnt how to read, we no longer see letters but words, and through education not only we no longer hear sounds but melodies, we also soon loose the capacity to hear a melody just as any melody, without perceiving it as classic music and then baroque music and then Handel's *Messiah* or our national anthem. Every step forward also means a loss. Human knowledge does not simply accumulates, it growths within a personal, social and cultural context, and it is always synthetic. There follows that, if an intelligent task can be successfully performed only on the basis of knowledge, experience, bodily involvement, interaction with the environment and social relations no alternative, non-mimetic approach is available, and any strong AI project is doomed to fail. To think otherwise, to forget about the non-mimetic and constructive requirements constraining the success of a computable approach, is to commit what we may call the Σ fallacy and believe that, since knowledge of a physical object, for example, may be in general described as arising out of a finite series of perceptual experiences of that object, then the former is just a short notation for the latter and can be constructed extensionally and piecemeal, as a summation.

Given the nature of LAI, it is not surprising that it has come to acquire a much less ambitious approach than GOFAI's. Indeed, the pragmatic approach shown by contemporary trends in LAI let one foresee that, rather than generating new forms of intelligence, AI is and will remain a fruitful source of technological innovations only insofar as human intelligence will be able to enlarge the number of processes that can be sufficiently “stupefied” to be carried on by a digital device. The failure of GOFAI is very far from entailing the impossibility of machines that can help us to think better, more efficiently and less expensively, or avoid us tiresome tasks. The major advantage of computer in most commercial and real-life applications lies in their remarkable capacity to perform routine functions quickly, endlessly and fairly reliably. The millions of computers working all over the world perform tasks that require rigid repetition of fixed sequences of instructions without complaint or fatigue, without intelligence of any sort. The view that I have tried to defend is that contemporary AI should be aiming at the creation not of a non-biological form of *Autonomous Intelligence*, but of an *Augmented Intelligence*, ours, and that future projects should be developed with an eye to such concept of a human, machine-aided intelligence. The failure of computational materialism has left us with the problem of explaining the genesis and nature of human intelligence and mental life, but it seems better and more honest to admit our ignorance and present lack of explanations than to endorse a wrong theory.