Ethics and Information Technology **6:** 175–183, 2004.

The responsibility gap: Ascribing responsibility for the actions of learning automata

Andreas Matthias
Computing Centre, University of Kassel, D-34109 Kassel, Germany
E-mail: matthias@hrz.uni-kassel.de

Abstract. Traditionally, the manufacturer/operator of a machine is held (morally and legally) responsible for the consequences of its operation. Autonomous, learning machines, based on neural networks, genetic algorithms and agent architectures, create a new situation, where the manufacturer/operator of the machine is *in principle* not capable of predicting the future machine behaviour any more, and thus cannot be held morally responsible or liable for it. The society must decide between not using this kind of machine any more (which is not a realistic option), or facing a responsibility gap, which cannot be bridged by traditional concepts of responsibility ascription.

Key words: artificial intelligence, autonomous robots, learning machines, liability, moral responsibility

Introduction

When people act, their actions have an impact on the lives of others; and so human societies, in the course of the centuries, have developed elaborate and differentiating rule systems in order to ascribe the responsibility for an action and its consequences justly.[1] When we judge a person to be responsible for an action, we mean either that the person should be able to offer an explanation of her intentions and beliefs when asked to do so, or that, following Strawson (1962), the person is *rightly* subject to a range of specific reactive attitudes like resentment, gratitude, censure, or praise (Oshana 2002: 263). For a person to be *rightly* held responsible, that is, in accordance with our sense of justice, she must have *control* over her behaviour and the resulting consequences "in a suitable sense" (Fischer and Ravizza 1998: 13). That means that the agent can be considered responsible only if he knows the particular facts surrounding his action, and if he is able to freely form a decision to act, and to select one of a suitable set of available alternative actions based on these facts.

Regarding the consequences of the operation of machines, we usually ascribe the responsibility for them to the operator of the machine, as long as the machine operates as specified by the manufacturer. The operator, by putting the machine into operation according to the manufacturer's specification, signals her acceptance of this responsibility. In case the machine does not operate according to the manufacturer's specification (that is, in case it has a flaw in its construction), we ascribe the responsibility to the manufacturer of the machine instead of the operator. This is in accordance with the principle mentioned above of *control* being a necessary condition of responsibility.[2] The operating manual of a device transfers control of that device from the manufacturer to the operator, by specifying the precise set of actions and reactions (in a system theory vocabulary: of *transformations*) the device is expected to undergo during normal operation,[3] thus enabling the operator

[1] For the purpose of this discussion, we will not usually need to distinguish between moral and legal responsibility (including liability). They will be dealt with together, and an explicit distinction will only be made where necessary.

[2] Although the notion of control used here is inspired by Fischer and Ravizza, we will not touch on the subject of determinism. This paper is about the practical problems which a society faces when it is trying to ascribe responsibility, and these are not altered by the determinism discussion, whatever its outcome may be.

[3] The *'operating manual'* may be quite an implicit affair when common artifacts of everyday life are concerned, for example, candles. There is (in the framework of a given society) a nearly universal understanding about how a candle is to be operated and what transformations it may undergo during operation. If, while using it properly, the operator burns down his house, then he himself is held fully responsible. If, on the other hand, the candle *explodes* while burning, then this behaviour is considered not to be part of the ordinary set of candle-usage transformations, and so responsibility is ascribed to the manufacturer instead, because the (implicit) set of operation instructions for a candle does not include provision for the case of an explosion, and thus the operator is bound to have *reduced control* over the device, which results in reduced (or absent) responsibility.

to handle it in a predictable manner, according to her own decisions on how to act.

In situations where the operator has reduced control over the machine he also bears less or no responsibility. If, for example a NASA technician operates from earth a remotely controlled Mars vehicle, and due to bad visibility in a Mars sandstorm and long response times (radio signals take 20 minutes to travel from Earth to Mars) the vehicle falls into a hole and is lost, then we would not consider the technician responsible for the loss. But who can be held responsible instead? In fact, *nobody*. In such cases of accidents that occur through no fault of a specific person, society refrains from ascribing responsibility, and collectively bears the cost resulting from the accident's consequences.[4]

In the following sections of this paper we will see how certain recent developments in the way of manufacturing computerised, highly adaptive, autonomously operating devices, inevitably lead to a partial loss of the operator's control over the device. At the same time, the degree in which our society depends on the use of such devices is increasing fast, and it seems unlikely that we will be able or willing to abstain from their use in the future. Thus, we face an ever-widening *responsibility gap*, which, if not addressed properly, poses a threat to both the consistency of the moral framework of society and the foundation of the liability concept in law.

Applications of learning automata

In order to provide the argument with some technical substance, let us look at a few examples: systems that are in development or already in regular use today.[5]

1. Let us first revisit the Mars vehicle case. Like the NASA *Pathfinder* (Morrison and Nguyen 1996), the vehicle will not only be controlled remotely from Earth, but it will also have its own integrated navigation and control system that enables it to avoid obstacles autonomously. Unlike *Pathfinder,* we will assume that the control program learns: after crossing a certain stretch of terrain, it will store into its internal memory an optical representation of the terrain as a video image, together with an estimate of how easy it was to cross that particular type of terrain. When a similar video image appears next time, the machine will be able to estimate the expected difficulty of crossing it, and it will thus be able to navigate around it if this seems desirable.
   Now let us assume that the vehicle again falls into a hole. Who is responsible? The operator obviously is not, since what caused the vehicle to disappear into the hole was not a manual control command, but a decision taken by the machine's control software itself. So, can the programmer be held responsible? He can deny responsibility by saying that the algorithm used was appropriate and correctly implemented. Surely one can assume that similar kinds of terrain will give similar video image representations. The actual decisions of the control program were based not only on preprogrammed data, but on facts that were added to the machine's database only *after* it reached the surface of Mars: they are not part of the initial program, but constitute genuine experience acquired autonomously by the machine in the course of its operation.
2. In high-rise office buildings, adaptive elevator systems already are used regularly.[6] These systems analyse traffic patterns, typically using artificial neural networks and reinforcement learning algorithms, and they try to minimise waiting and transportation time for the users of the elevator. Now let us suppose, that such a learning, adaptive elevator leaves an important executive waiting for half an hour in the 34th floor, so that he cannot attend a business meeting at which he is expected. A considerable financial damage is caused. Who can be held responsible?[7] The manufacturer can deny responsibility because the elevator, being able to learn, had changed the parameters of its program during the course of its operation, so as to better adapt to the traffic patterns in the building. Because of this, it was no longer possible for the manufacturer to predict or control the specific behaviour of the elevator in a given situation (though it might be possible to prove mathematically that eventually the used algorithm will

[4] Taxation and insurances are some of the ways society has devised to distribute the cost of such accidents to a broad base of its members.

[5] There are many other such systems in use, far too numerous to be dealt with here explicitly, including: electronic noses for banana ripeness determination (Llobet et al. 1999), collision avoidance systems in automatic submarine navigation (Schultz 1991), autonomously flying (Stancliff and Nechyba 2000), game playing (De Jong and Schultz 1988), and web document filtering (Zhang et al. 2000) machines and programs.

[6] OTIS (2003), Schindler (2003), Sasaki et al. (1996).

[7] The case is legally more or less interesting, depending on product liability laws in different countries and legal traditions. But even where no legal liability can be identified (for example in German law), it is possible to ask who *should* (on moral grounds) be considered responsible for the damage done by the machine.

converge to some optimal behaviour, that is, minimal overall waiting time for the users of the elevator). And since the manufacturer could not have predicted or averted the undesirable outcome in this case, he cannot justly be held responsible for this specific behaviour of the machine.[8]

3. Zhou et al. present a system for the automatic diagnosis of lung cancer. The system learns to identify cancer cells on microscope images of specimens of needle biopsies obtained from the bodies of the persons to be diagnosed. Systems like that are supposed to be used where an experienced senior pathologist is not available (e.g., in underdeveloped rural areas). The system has been constructed so that false negative diagnoses are highly improbable (proclaiming the patient to be healthy when there are, in reality, cancer cells present), but there is accordingly much less precaution about false positiva. Although wrong positive diagnoses are not immediately life-threatening, they can cause great financial, practical and emotional problems to the affected parties, and so again we must find someone to ascribe responsibility to for such erroneous diagnoses of the machine.[9] As far as the programmer is concerned, he has done everything possible to prevent false negative diagnoses, which is the best he could do,[10] so he can also not justly be held responsible if the limitations of the machine are clearly known beforehand.[11]
4. One last example: since 1999, there has been a mobile robot distributed by Sony ('AIBO'), which is supposed to be an intelligent toy for children as well as a pet replacement for the small urban apartment. This machine is capable of learning: in Hornby et al. (1999) (and many other publications) we see how AIBO learns new things: words to which it is to react ('come here'), but also new kinds of movements, for example when the built-in stepping algorithm proves not to be optimal for the crossing of deep persian carpets.[12] Let us now assume that an advanced version of the robot is able to change its walking style with time to optimise overall performance. With a little experimentation it will be able to find out that its battery life can be prolonged by galloping, which reduces the friction between itself and the ground. Let us also assume, that the robot, while running around the apartment, collides with a small child and injures him. Who is now responsible? The manufacturer? Why exactly? The child's parents for putting the pet robot into operation in their apartment? Or is this an *unforeseeable* development, which occurred due to the adaptive capabilities of the robot, so that nobody can be justly said to be responsible?

Let us summarise the points: presently there are machines in development or already in use which are able to decide on a course of action and to act without human intervention. The rules by which they act are not fixed during the production process, but can be changed during the operation of the machine, *by the machine itself*. This is what we call machine learning. Traditionally we hold either the operator/manufacturer of the machine responsible for the consequences of its operation, or 'nobody' (in cases, where no personal fault can be identified). Now it can be shown that there is an increasing class of machine actions, where the traditional ways of responsibility ascription are not compatible with our sense of justice and the moral framework of society because nobody has enough *control* over the machine's actions to be able to assume the responsibility for them. These cases constitute what we will call the *responsibility gap*.

In order to fully understand the nature and implications of this problem, it will be necessary to have a look at how learning automata are being constructed.

[8] He might nevertheless be held responsible for selling elevators of this type, if the dangers of using them are not advertised properly in the operating manual. But this is a different point, and, as we will see later on, there might not have been any choice of an alternate, 'safe' system at all.

[9] The trivial solution would be to insist that a human expert should verify the correctness of the machine-made diagnoses and assume the responsibility for them. But this is obviously not possible, because the machine was developed exactly for those situations where a human expert is not available. Were it possible to supply every machine with a controlling human expert, nobody would need the machine in the first place.

[10] When in doubt, the system will always assume that a cancer cell is present. There is no unfailing machine (as there is no such human) and the programmer has (correctly) decided to shift the probability of an error into the less dangerous realm of false positive diagnoses instead of false negative ones.

[11] Here the society as a whole decides to employ an inherently risky technology, because it is assumed that the overall benefits of using it outweigh the risks.

[12] Some of these experiments require modified versions of the robot that are not available on the market, and so does our thought experiment. But this is not relevant to our discussion, since the required modifications can be done easily, at least by the manufacturer himself (and have already been done for experimental purposes).

Engineering learning automata

This is, of course, not the place to give an introduction to artificial intelligence.[13] We will only look at those specific properties of artificial learning systems which are of importance to the question that concerns us, and so, the following exposition will be kept very brief.

We can distinguish four primary types of learning automata which are of interest in the present context: symbolic systems, connectionist architectures (including reinforcement learning systems), genetic algorithms (including genetic programming), and autonomous agents (mobile and immobile).

Symbolic systems

Symbolic systems are built on the assumption that some of the cognitive functionality of humans which we call intelligence can be expressed as a syntactic manipulation of linguistic symbols.[14] A system built along these lines contains long sequences of axioms and derivation rules that are usually expressed in some kind of predicate calculus. The system is either able to derive conclusions from the facts stored in its database, or else to extend this database by adding new rules and facts to it (which is what constitutes learning).[15]

The symbolic approach to artificial intelligence has a long tradition, and it has led to practically usable programs: expert systems that are able to advise users and provide solutions in specific, clearly defined application domains. Classical examples of expert systems include medical diagnosis programs and fault identification in complex technical systems (see Figure 1).

IF:	The exhaust is smoky, and The car is backfiring, and There is a lack of power
THEN:	The carburetor fuel mix is too rich.
IF:	There is a lack of power, and There is a gray deposit on the spark plugs, and The engine overheats,
THEN:	The carburetor fuel mix is too weak.
IF:	The carburetor fuel mix is too rich, or The carburetor fuel mix is too weak,
THEN:	The carburetor fuel mix needs to be adjusted.

Figure 1. Example of an expert system rule base: some rules used in the diagnosis of motor faults (after Andriole and Hopple 1992).

For our present discussion it is important to note that the knowledge of expert systems (and therefore also the actions of such systems, which are based on that knowledge) are stored inside the system in the form of explicit, distinct, quasi-linguistic symbols. They can be inspected at any time and, should need arise, be corrected.

Connectionism and neural nets

While symbolic artificial intelligence presupposes the existence of clear and distinct symbolic representations of objects and the relations between them, connectionism does not. Instead, it attempts to emulate the basic principles of neural operation in living systems. It is based on the observation that biological information processing systems do not seem to represent symbols as discrete entities, but distributed all over the neural net. Information is stored by modifying the architecture of the network and the strength of individual connections between neurons (represented as input 'weights' in artificial networks; see Figure 2). During learning, the network adapts these weights in order to minimise the difference between the actual and the expected output for a given learning pattern. It is essential to this concept that there nowhere is a 'list' or 'catalog' of all learned information, as there is in symbolic programs. Symbolic programs contain axioms, predicates, facts and rules, which represent information *explicitly* and *verifiably*. Connectionist systems lack an explicit representation, and the contained information can only be deduced from their behaviour. Neural networks are, owing to the principles of their architecture, black boxes. We can evaluate their behaviour by applying test patterns and observing their output, but we cannot

– have a look at the information that is stored inside the network, and, even more importantly;

[13] Although we will occasionally use the term 'artificial intelligence' to refer to an academic field of study, we will not enter the discussion as to whether it constitutes "real" intelligence or not. The discussion about the nature of intelligence and if artificial systems can possibly be intelligent is not relevant to the question at hand, since the responsibility gap concerns the *human* part of the human–machine relationship.

[14] These need not necessarily be symbols of a natural language. Often symbolic AI systems incorporate custom formal languages, in which the system's knowledge of the world can be expressed in some way that is useful to the system.

[15] These processes can get quite complex, including programs that derive conclusions by deduction, induction, reasoning by similarity and analogy, conflict resolution between facts in the database, and probabilistic or fuzzy reasoning in the face of unsure facts.

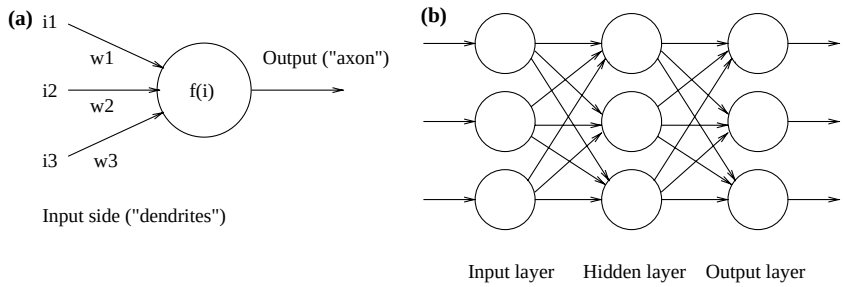

Figure 2. Neural networks.(a) The basic structure of a single artificial neuron. The input signals (i1–i3) are multiplied with (changing) weight factors (w1–w3) and then used as parameters to a function that calculates the output of the neuron. During the learning process these factors change, thus modifying the neuron's behaviour. (b) In a basic backpropagation neural network there are multiple layers of neurons which are interconnected, so that the output of one layer serves as input to the next.

– see what information is *not* represented inside it.[16]

The behaviour of neural networks is not programmed in the way a procedural program is. Instead they are trained *by example:* sets of input patterns are presented to the network and the internal weights are changed gradually, so that the network eventually produces a desired, corresponding set of output patterns. It is not necessary for the trainer of a neural network to be able to express the information to be learned in the form of clear and distinct symbols, so that finally neural networks can successfully learn to make distinctions for which the human trainer himself is unable to provide an algorithmic representation.[17]

Reinforcement learning (Moriarty et al. 1999) lifts the distinction between the training and application phases which we find in traditional neural network concepts. The system learns inside its final operating environment by exploring available action alternatives in a trial-and-error fashion and optimising its own parameters according to the results. Thus, the exploration phase is an integral part of the design of the working machine and cannot be separated from it. This is necessary in highly dynamic environments, where a system needs to change continually in order to achieve optimal efficiency. Consider, for example, a high-rise office tower with ninety floors and a sixteen elevator system which is controlled by software. This system cannot work optimally if it is not always learning and adapting to the changing traffic flows in the building. Imagine there is, for the duration of one week, a conference on floor 34. The system must be able to optimise its behaviour during this time, for example by leaving idle elevators waiting at this floor. After the conference is over, however, this behaviour will have to change again to accommodate new needs. This can only be achieved with systems which learn in their respective operating environments. And since learning is done by trial and error, we find that *errors are unavoidable features of any reinforcement learning system*. This is quite contrary to the common, traditional understanding that technology, done correctly, must operate free of errors, and that errors are always the errors of the programmer, not of the programmed machine. Reinforcement learning presents us, for the first time, with the *necessity* of errors as we know it from living systems: as a system *feature,* the precondition for learning and adaptive behaviour, and not merely a product flaw.

Genetic algorithms

While neural networks are modelled after the neural infrastructure of living systems, *genetic algorithms* (Holland 1975) imitate the principle of evolution through variation, genetic recombination, and selection. In a genetic algorithm system the solution to a problem is typically represented by an ordered chain of symbols that are selected out of an 'alphabet'. Think, for example, of a program that is supposed to find its way through a maze. A solution to the problem of traversing the maze would consist of a sequence of direction change commands: 'right,' 'left,' 'left,' 'right,' and so on, which together describe the way from the entrance to the exit of the maze (see

[16] All the social procedures that humans and animals have developed to evaluate other individuals (rituals of introduction, teaching by demonstration, examinations and psychological tests) are attempts to solve this problem: that we cannot look into the heads of others and *see* the knowledge, intentions and beliefs stored therein. Instead we are forced to infer them indirectly through observation of the other's behaviour and the application of test patterns.

[17] This is especially true when we consider noisy or distorted input, as in the case of optical character recognition, which is very hard to do algorithmically, but almost trivial with artifical neural networks.

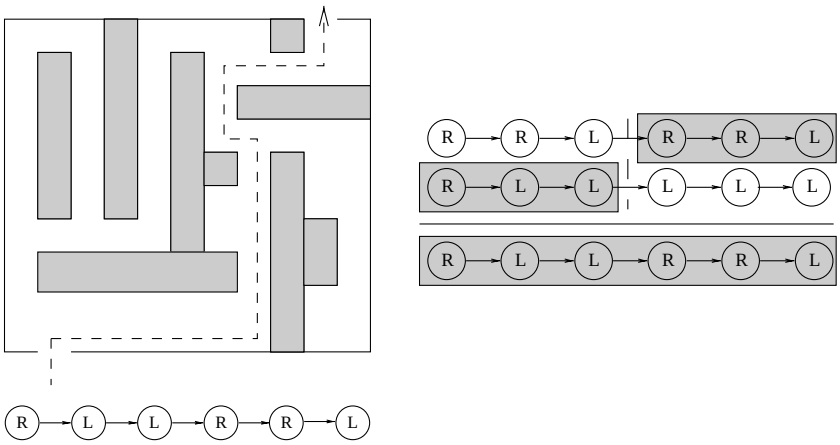

Figure 3. The way through a maze and its representation in a quasi-genetic 'alphabet' of direction-change statements (left). The chain of left–right-statements (the 'genetic material') is manipulated with operations like the cross-over shown here, so that eventually the algorithm arrives at a solution (right).

Figure 3). In the world of the genetic algorithm metaphor, we describe this sequence as the 'genome', which is composed of directional genes with the alleles 'right' and 'left'. The search for a solution begins with a big population of virtual organisms which are initialised with a random genome, that is, with a random sequence of direction-change statements. Limiting factors in the simulated environment provide the required selection pressure: if an organism has been moving around unsuccessfully for a while, or has collided too often with the walls of the maze, it is removed from the simulation (it 'dies') and its genetic information is lost. Successful organisms (that move forward without colliding with the walls or moving in circles) are given the chance to 'reproduce'. During reproduction the genetic material of one or more parental organisms is inherited by the filial organisms.[18] After a while, the simulation produces organisms which can cross the maze successfully (though the solutions found are not necessarily optimal).

Genetic algorithms can be best applied to problems which solutions are representable as (not overly long) chains of symbols, and for which the solution can be approximated gradually by trial and error. Some applications are the evolution of behaviours for mobile robots (Schultz 1994), the classification of sensoric input (Stolzmann et al. 2000), and the navigation of autonomous vehicles (Schultz 1991).

What we have seen just now could be addressed as the ''symbolic'' variant of genetic algorithms. But it is also possible to combine genetic algorithms with connectionist architectures, by *evolving* neural nets, instead of defining their architecture by hand (Belew, et al 1990; Nolfi and Parisi 1991).

Genetic programming still goes one step further. Here, the result of the simulated evolution is not the solution vector itself any more, but a *software program* that implements the solution. Thus, the genetic algorithm acts itself as a programmer. We will address the significance of this fact in the context of responsibility ascription later in this paper.

Autonomous agents

Finishing this brief survey of artificial intelligence programming methods, we will have a parting look at the technology of artificial agents. Autonomous agents are artificial entities that fulfil a certain, often quite narrow purpose, by moving autonomously through some 'space' and acting in it *without human supervision*. The agent can be a software program that moves through information space (e.g., an internet search engine spider), but it can also have a physical presence (e.g., a computer-driven vacuum cleaner or a robotic pet) and move through the space of an apartment.

Two points concerning agents are interesting in the present context: First, that agents are *per definitionem* designed to act, and that, in the course of their operation, they must inevitably interact with other things, people, and social entities (laws, institutions and expectations). Second, that agents which have a physical presence constitute a new category of machines: such that can learn from the direct interaction with a real environment and that can in return directly manipulate this same environment.

[18] The specific rules of reproduction and genetic material transfer vary widely between different implementations of the genetic algorithm, but this makes no difference for the purpose of this paper.

These machines have (contrary, for example, to desktop computers) an unmediated access to sensory impressions, their symbolic representation, and subsequent actions that lead to new sensory impressions, and they act in the same environment as humans do.

The responsibility gap

Let us now return to the central question of this paper.

To what extent is it possible to hold the manufacturer/programmer/operator of an autonomous, learning automaton responsible for the actions of the machine?

If we look at the role of the machine's manufacturer (or, in our case, the programmer) and the change of this role, from the straightforward programming concepts of traditional software engineering (see Figure 4), to the construction of autonomous artificial intelligence systems, we can clearly see the profound change that has taken place in the past few years. In the beginning we find the programmer as *coder,* that is, someone who expresses the program (and thus the operating behaviour of the machine) line by line and statement by statement in a linguistic representation that can be executed directly by the machine (the statements of a programming language). At any moment, it is possible to inspect the memory of the machine and to determine the precise extent of the program. Changes can be made directly through the addition, removal or exchange of particular statements in the program. The programmer, *in control* of the behaviour of the machine in every single detail, can explain the way his algorithm works, and an observer can follow this explanation and check its correctness. Errors are always errors of the *programmer,* not errors of the *program*. They can always be identified, isolated and fixed, and the programmer can rightly be held responsible for any misbehaviour of the machine.

As the techniques of artificial intelligence programming develop further, the role of the programmer changes. This change is not sudden but gradual, and its extent differs according to the technology employed.

It begins with the programmer who uses logic-oriented programming languages and symbolic expert systems losing control over the execution flow of the program. Programming languages based on predicate logic are not executed in the same linear fashion as their procedural, imperative counterparts: instead of being executed from the first to the last 'command,' they use a run-time system which searches (in a way not always known to the programmer) for deductions from given axioms and inference rules. The flow of control in such systems is typically difficult to describe and should, ideally, be of no interest to the programmer. This is, in itself, not a big problem because as long as there is a symbolic representation of facts and rules involved, we can always check the stored information and, should this be necessary, correct it.

In artificial neural networks, the symbolic representation of information and flow control disappears completely: instead of clear and distinct symbols we have a (possibly very large) matrix of synaptic weights, which cannot be interpreted directly any more. The knowledge and behaviour stored in a neural network can be only inferred indirectly through experimentation and the application of test patterns after the training of the network is finished (as is the case with living systems).

```
FilenameStruct * readdirectory( FILE *fp, int *count) {
          FilenameStruct *fn;
          int i, len = GetWord(fp);

          *count = (len / 17) - 1;
          fn = (FilenameStruct *) malloc(*count * sizeof(FilenameStruct));
          for (i = 0; i < *count; i++) {
               fn[i].offset = GetLong(fp);
               fread(fn[i].fname, 13, 1, fp);
               fn[i].fname[13] = 0;
               lowerstr(fn[i].fname);
               if (showdirectory)
                    fprintf(stderr, "%13s %6d\n", fn[i].fname, fn[i].offset);
               }
          return (fn);
     }
```

Figure 4. Part of a program in the 'C' programming language. The programmer of such a *procedural* or *imperative* program must enter every single program statement herself. All program statements together implement an algorithm for the solution of the problem at hand.

Reinforcement learning, being usually based on neural network concepts, shares the same problems, but additionally it lifts the distinction between a training and a production phase. Reinforcement learning systems explore their action space, while working in their operating environment, which is their central feature (enabling them to adapt to ever-changing environments) as well as a big drawback concerning their predictability. The information stored in the network cannot be fully checked, even indirectly, because it always changes. Even if we can prove mathematically that the overall performance of such a system will eventually converge to some optimum, still there will be *unavoidable* errors on the way to that optimised state. The creator of such a system (not really being a programmer in the traditional sense any more) cannot go about eliminating errors; instead he must explicitly permit them, so that the system can stay operational and improve its performance.

An additional layer of obscurity comes with the use of genetic programming methods. Here, the result of the simulated evolution is not the solution itself, but a program that implements the solution. Thus, we get an additional layer of machine-generated code that gets between the programmer and his product. While in neural networks the designer still defines the operating parameters of the system (the network architecture, the input and output layers and their interpretation); and while with genetic algorithms at least he can define the alphabet used and the semantics of its symbols; with genetic programming he loses even this minimal amount of control, and he creates a machine that programs itself.

Finally, autonomous agents also deprive him of the spatial link between him and his product. The agent acts outside the observation horizon of its creator, who, in the case of a fault, might be unable to intervene manually (because he might not know about the fault until a much later point in time). This is the case for pure information agents (internet indexing programs) as well as for agents which have a physical manifestation (autonomous space vehicles, targeting systems of military missiles, mobile electronic pets).

Thus, we can identify a process in which the designer of a machine increasingly loses control over it, and gradually transfers this control to the machine itself. In a steady progression the programmer role changes from *coder* to *creator* of software organisms.

In the same degree as the influence of the creator over the machine decreases, the influence of the operating environment increases. Essentially, the programmer transfers part of his control over the product to the environment. This is particularly true for machines which continue to learn and adapt in their final operating environment. Since in this situation they have to interact with a potentially great number of people (users) and situations, it will typically not be possible to predict or control the influence of the operating environment.[19]

Conclusion

We presented some of the reasons why certain classes of autonomously adaptive machines must inevitably display suboptimal or even outright erroneous behaviour; behaviour that must be attributed to the machine itself and not to its designer or operator.

1. In the course of the progression of programming techniques: from the conventional procedural program, via neural network simulations, to genetically evolved software, the programmer loses more and more of her control over the finished product. She increasingly becomes a 'creator' of 'software organisms', the exact coding of which she does not know and is unable to check for errors.
2. The behaviour of the machine is no longer defined solely by some initial, and henceforth fixed program, but increasingly shaped by its interaction with the operating environment, from which the machine adapts new behavioural patterns that constitute solutions in the machine's problem space.
3. In order to be able to adapt flexibly to new situations (which is necessary if they operate in dynamic and changing environments), automata must leave behind the clear separation between programming, training, and operation phases. Practically useful technologies will have to learn during operation, which also means that they will have to make 'mistakes' during operation (a "mistake" being just the exploration of the solution space by the machine itself, which enables it to arrive autonomously at new solutions).
4. There are an increasing number of situations in which the supervision of an operating machine by a human expert is either in principle or for economic reasons impossible. The supervision of an operating machine is impossible when the machine has an informational advantage over the operator (e.g., navigation computers in cars, radar-based

[19] Except statistically, but a statistical fact (97% of elevators operate within 1% of their optimal performance) is not a guarantee against errors of the machine, and therefore not a useful answer to the responsibility question.

flight control systems, a lung-cancer diagnosis machine in a remote rural area). It is also impossible when the machine cannot be controlled by a human in real-time due to its processing speed and the multitude of operational variables involved (e.g., the control system for a group of 16 elevators in a 90-floor building, or the controlling computer for a nuclear power plant).

5. Still, we cannot do without such systems, because the pattern processing and systems control tasks that we must accomplish in our highly dynamic and complex environments are so complicated that they cannot be addressed by simpler, statically programmed machines. Planetary exploration, traffic control, automated medical diagnosis: these are tasks that cannot be solved by simple, verifiable algorithms. Not to address such tasks seems not to be an acceptable option to most citizens today.

Automatic machines leave their traditional operating environments, and increasingly move into problem areas that, owing to their dynamic nature and complexity, have previously only been accessible to humans. It is natural that in the course of this transition they will not be able to avoid also acquiring some of man's limitations (e.g., learning from experience as the basis of flexible behaviour; experience being just another word for potential error.)

If we want to avoid the injustice of holding men responsible for actions of machines over which they *could not have* sufficient control, we must find a way to address the responsibility gap in moral practice and legislation. The increasing use of autonomously learning and acting machines in all areas of modern life will not permit us to ignore this gap any longer.

References

S.J. Andriole and G.W. Hopple. *Applied Artificial Intelligence: A Sourcebook*. McGraw-Hill, New York, 1992.

R.K. Belew, J. McInerney and N.N. Schraudolph. Evolving Networks: Using the Genetic Algorithm with Connectionist Learning. Cognitive Computer Science Research Group, Computer Science & Engineering Department (C-014), University of California at San Diego. *CSE Technical Report #CS90–174,* June, 1990.

K.A. De Jong and A.C. Schultz. Using Experience-Based Learning in Game Playing. *Proceedings of the Fifth International Machine Learning Conference,* (pp. 284–290). Ann Arbor, Michigan, June 12–14, 1988.

J.M. Fischer and M.S.J. Ravizza *Responsibility and Control. A Theory of Moral Responsibility*. Cambridge University Press, Cambridge, 1998.

J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.

G.S. Hornby, M. Fujita, S. Takamura and others. *Autonomous Evolution of Gaits with the Sony Quadruped Robot*. Group 1, D-21 Laboratory, Sony Corporation Ph: 81-3-5448-5901. Tokyo, Japan, 1999.

E. Llobet, E.L. Hines, J.W. Gardner and S. Franco. Non-Destructive Banana Ripeness Determination Using a Neural Network-Based Electronic Nose. *Measurement Science & Technology* 10: 538–548, 1999.

D.E. Moriarty, A.C. Schultz and J.J. Grefenstette. Evolutionary Algorithms for Reinforcement Learning. *Journal of Artificial Intelligence Research,* 11: 199–229, 1999.

J.C. Morrison and T.T. Nguyen. On-Board Software for the Mars Pathfinder Microrover. *Jet Propulsion Laboratory report IAA-L-0504P,* 1996.

S. Nolfi and D. Parisi. Growing Neural Networks. *Institute of Psychology, National Research Council Technical Report PCIA-91-15,* Rome, Italy, 1991. (Presented at Artificial Life III, Santa Fe, New Mexico, June 15–19, 1992.)

M.A.L. Oshana. The Misguided Marriage of Responsibility and Autonomy. *The Journal of Ethics,* 6: 261–280, 2002.

OTIS Elevators. Redefining Elevator Performance, Safety and Comfort: The OTIS Elevonic Class. Product Description, 2003. Available: http://www.otis.com

K. Sasaki, S. Markon and M. Makagawa. Elevator Group Supervisory Control System Using Neural Networks. *Elevator World,* 1, 1996.

Schindler Elevator Corporation AITP: Artificial Intelligence Traffic Processor. Technical Product Description, 2003. Available: http://www.us.schindler.com

A.C. Schultz. Using a Genetic Algorithm to Learn Strategies for Collision Avoidance and Local Navigation. In *Proceedings of the Seventh International Symposium on Unmanned Untethered Submersible Technology,* (pp. 213–215). University of New Hampshire Marine Systems Engineering Laboratory, New Hampshire, September 23–25, 1991.

A.C. Schultz. Learning Robot Behaviors Using Genetic Algorithms. In. *Proceedings of the International Symposium on Robotics and Manufacturing*. Washington DC, August 14–18, 1994.

S.B. Stancliff and M.C. Nechyba. Learning to Fly: Modeling Human Control Strategies in an Aerial Vehicle. Machine Intelligence Laboratory, Electrical and Computer Engineering, University of Florida, 2000. Available: http://www.mil.ufl.edu/publications.

W. Stolzmann, M.V. Butz, J. Hoffmann and D.E. Goldberg. First Cognitive Capabilities in the Anticipatory Classifier System. *Illinois Genetic Algorithms Laboratory Report No. 2000008,* University of Illinois, Urbana, 2000.

P. Strawson. Freedom and Resentment. *Proceedings of the British Academy*, 48, 1962.

B.-T. Zhang and Y.-W. Seo. Personalized Web-Document Filtering Using Reinforcement Learning. AI Lab, School of Computer Science and Engineering, Seoul National University, Korea, 2000. Available: http://www.scai.snu.ac.kr

Z.H. Zhou, Y. Jiang, Y.B. Yang and S. F. Chen. Lung Cancer Cell Identification Based on Artificial Neural Network Ensembles. *Artificial Intelligence in Medicine,* 24(1): 25–36, 2002.