Anticipation, Induction, and Learning

Bertil Ekdahl
Lund Institute of Technology
School of Engineering
Box 882, SE-251 08 Helsingborg, Sweden
Fax: +4642176337 - E-mail: Bertil.Ekdahl@cs.lth.se

Abstract

A system is considered anticipatory if it has the ability to foresee the consequences of an event and act in a way it is adapted for. In order to make such judgments anticipatory systems must possess some kind of description of their surroundings, which is used in the calculation of an appropriate action. In many cases it is sufficient to have an algorithmic description to follow and some anticipatory systems do choose their actions in a completely algorithmic way.

A more developed anticipatory behavior is displayed by systems, which not only possess a description but also a model of the surroundings. Those systems have an intrinsic conception of their surroundings, which they are able to reason about. This kind of anticipation is called model-based contrary to the description-based behavior, which characterizes those systems that slavishly follow algorithmic rules.

In order to take advantage of model-based behavior it is necessary to be able to properly describe the surroundings in terms of how they are perceived. Such description processes are inductive and not recursively describable. That a system can perceive and describe its own surroundings means further that it has a learning capability. Learning is the process of making order out of disorder and this is precisely the most distinguish quality of inductive inference. Genuine learning without inductive capability is impossible.

The implication of this is that systems that have a model of the surroundings are not possible to implement on computers nor can computers be learning devices contrary to what is believed in the area of machine learning.

Keywords: Anticipation, Anticipatory systems, Induction, Learning, Machine Learning.

1 Introduction

In a certain sense most systems can be considered causal because of their behavior being a consequence of an outer cause. Even organic systems (living systems) can by evolution be considered developed by cause but the way living systems react to a cause is in most cases quite different from the way non-living systems react. All non-

living systems react in the same way on a given cause contrary to living systems whose reaction on one and the same cause can vary widely between different species and even among individuals within a species. The reason that living systems react differently than the non-living is of course that the former are subject to the evolutionary process acting on phenotype to which there is no counterpart in the systems studied in physics. An outer pressure (circumstance) does not change the rules that govern those systems: the rules remain the same (even if the way we describe them may be refined and more accurate).

Instead of the mere cause it is the ability or lack of ability to choose action on a given or expected event that normally categorizes systems. If a system lacks the means to influence its own reaction on a future event, it is usually called *causal* or, as will be used here synonymously, *reactive*. Here I refer to causality as it is used in physics as explained for example by D'espagnat (1999, p. 94): "[...] two familiar milestones still standing erect. One is the possibility of keeping a nonrelativistic *causality principle* in the technical sense of the term: no property at a time t is determined — or even affected — by the events that may occur thereafter." Systems that, contrary to the reactive, can foresee the outcome of a cause and act appropriately in order to avoid unwanted consequences are called *anticipatory*. In the technical sense it means that the behavior of anticipatory systems at a certain moment is affected by future possible events, or in other words, may take future events into consideration in the current situation. As an elucidation of the two principles we may consider a marble moving against a wall. The marble, as a consequence of the causality principle, has no possibility of itself to avoid hitting the wall. Had the marble been anticipatory it would foresee that it was going to be broken into pieces and accordingly act in order to avoid this harmful consequence.

An outside observer cannot explain the behavior of an anticipatory system without knowing the system's intrinsic judgment of how its surroundings will affect it. Evidently, in order to make such a judgment anticipatory systems must possess some kind of description of their surroundings, which is used in the calculation of an appropriate action. Reactive systems do not possess such descriptions. In those cases the description is solely in the mind of the observer. This makes it possible to categorize systems from a linguistic point of view: systems with and without an inner description.

Anticipatory systems, which have an inner description, can further be divided into two disjunct classes. One that consists of those systems that possess only a description without knowing its interpretation and another consisting of those having a model in the semantic sense. Ekdahl (1977) has classified them as *description-based* and *model-based* anticipatory respectively due to their different linguistic capability. The characteristic of the description-based anticipatory systems is that they have a definite description to follow. Such a description cannot be partial since it should imply that there were situations for which there was no description. Accordingly, it means that the description to follow is algorithmic and consequently that there is no way going outside the description. This is true even if we think about a self-modifying program (algorithm). From a computational standpoint, we can imagine such a program as

computed by a universal Turing machine that first simulate the behavior of the Turing machine x (x is the index of the machine or, equivalently, the program coded as a Gödel number) with input u, written as $[x](u)$, and when that execution is finished, change to the behavior of the Turing machine $[y](v)$. (See Papadimitriou (1994) for a good treatment of Turing machines.) Here the programs $[x]$ and $[y]$ are both supposed to be total since our starting point was an algorithm. Despite changing the program, this is still a description to follow that is decided from the beginning by the programmer and always terminate and there is no possibility for the universal Turing machine to compute another algorithm than $[x]$ and $[y]$ in that order. Thus, having an algorithmic description to follow, the description-based anticipatory systems have no explicit idea of the world around them, as is the case for example for cerebral systems. Contrary to description-based systems, the model-based have models in the semantic sense and are able to reason about their models (in a metalanguage to the model) and have also the ability to change the model if necessary. A planning being is an example of a model-based anticipatory system while an insect can serve as an example of an anticipatory system that is description-based.

In this paper it is argued that in order for a model-based anticipatory system to be able to produce descriptions on its own based on its model of the surroundings it has to have an inductive capability. The description-based systems, without this capability, are obedient to follow an algorithmic behavior and can by no means change their description in order to adapt it to changes in the surroundings.

It is also shown that learning and induction are intimately connected and systems without inductive ability have no genuine learning ability. They can generate conclusions only from already established premises and these premises are programmed, not acquired by experience. This has the implication that model-based anticipatory systems cannot be implemented on computers and furthermore, that genuine machine learning is in fact impossible.

2 Learning and Knowledge

The terms *learning* and *knowledge* are intimately connected but, as used in everyday language, neither is a very concise concept. With learning we always associate a moment of new knowledge and it is exactly what should be meant by new knowledge that is crucial when we try to formalize learning. The normal way to think about the two concepts is that when you learn something you increase your knowledge: what one learns one did not know before. It is in this sense we may consider a learning device as a device, which increases its knowledge, i.e., after the learning, the device, in a sense, *knows* more than before.

The most genuine case of learning is when we have explored from the environment something we did not know before. Here the subject matter of learning is the empirical world, which interact with the learning device. An example of such learning is the generalization of observations leading from an existential statement to a universal statement as "*all P are Q*". To such a conclusion we arrive at by induction,

i.e., an inference that generalizes a few observations about members in a set to all members in the set. By way of an example, suppose we notice that all observed ravens are black. If it seems reasonable to infer that "all ravens are black" then it is an inference arrived at by induction. To go from the proposition "*some P are Q*" to the conclusion "*all P are Q*" is a description of order in our environment. What is observed and described is primarily the underlying regularity, which is extracted to such an extent that predictions can be made. Science is essentially such a learning device (Hesse, 1973) where regularities and order in nature are formulated in descriptions in the form of universal statements.

To be able to describe a phenomenon we must have a suitable collection of predicates formulated in an already existing language. The phenomenon in question must already be known and perceived as existent but the order, created by the hypothesis, is not known in advance. By way of an example, there was a time when witches populated Europe. Witches were believed and there were a mental readiness for perceiving them. As time passed and we learnt more about the world it was realized that there are no such entities as witches. That is to say that knowledge of the world is much a consequence of our existential perceptions. We do not know the world; we have to create it in terms of existential objects. Compare Bridgeman's (1936) operational view on existence:

> [..] *how do I know that the tables, the clouds, and the starts of ordinary experience exist? These are not given directly in experience, but are constructions. It seems to me that the answer to all these questions is that the thing exists because for one thing the concepts work in the way I want them. In my effort to solve the problem of adapting myself to my environment I invent certain devices, and some of them are successful and I use them in my thinking.*

The idea of associating learning with extraction of regularities and order is not new and can be traced back to the Gestalt school in Germany. According to this also insight is a kind of learning.

In science, descriptions arrived at by induction are called *hypotheses*. A hypothesis with predictive power is called a *theory* when it is sufficiently confirmed. There is a widespread misuse calling a hypothesis a theory despite the lack of predictive power. As an example, the *theory of* evolution is a well-confirmed hypothesis but, despite being called so, it is not a theory since it cannot predict how changes will occur, just that it may occur.

Since knowledge of the world depends on how one perceives the world, learning also involves change of knowledge. The change of knowledge is necessary when new phenomena are impossible to describe in the current language or when an old belief no longer agrees with current conception. An example of the former is quantum phenomena, which were not possible to describe in the language of classical mechanics since quantum phenomena were not consequences of the classical theory. The disbelief of witches is an example of where old ideas do not fit in a new conception of the world.

We also talk about learning in quite a different context, namely when we obtain a new mathematical theorem or something else that we conclude by pure deductive reasoning. This kind of learning is not of the same quality as induction since a theorem is a consequence of axioms in a formal system. No new knowledge is created, the axioms are the same as before and thereby the theory. If we consider theories with a recursive set of axioms, the set of theorems is recursively enumerable and there is a recursive function that can generate all theorems in the theory. This derivation of theorems is completely mechanical and can be done by a computer (Turing machine). The equivalence between recursive functions and computing machines is a consequence of Church's thesis. Making a derivation does not involve any understanding of the symbols used, contrary to induction where conclusions are about concepts, i.e., such entities to which symbols are interpreted.

For people, as a learning device, the two ways to look at learning do not involve any problem since in the term *learning* there is no restriction to a special method. In many cases learning means *understanding* and in this respect deductively generated conclusions can be regarded as new knowledge. For example, when someone learns *Pythagoras' theorem* and understands it to such an extent that the whole idea behind the theorem can be grasped, it is reasonable to talk about new knowledge. However, when formal systems come into consideration, the question arises whether a consequence, deductively inferred, can be considered new knowledge.

3 The Learning Process

Induction, as a learning method, is an information creating process in the way that the result cannot be inferred from the background knowledge. Instead there is an increase in knowledge that stems from the inductive process. A good example from physics is Max Planck's discovery of the quantum of action, which inaugurated a new epoch in the physical science. Pais (1991, p. 86) has the following description of Planck's way to find his law.

> Experiment had given him every reason to believe that he had the correct spectral density ρ. From ρ he could read off the average resonator energy U. Simple thermodynamics had led him from U to the resonator entropy S. In turn S is related to probability W by Boltzmann's principle. Problem: To find and interpret the expression for the fundamental probability W which, by arguing backward, yields the known ρ: from W to S to U to ρ.

As is evident, this process of reasoning is not deductive since contrary to such a process we are searching for the premises from the conclusion. This procedure is typical for induction. Wang, (1987, p.180) has explained this process in the following way:

> We tend to believe the premises [axioms] because we can see that their consequences are true, instead of believing the consequences because we

> know the premises to be true. But the inferring of premises from consequences is the essence of induction [...]

The process of induction aims at a universal description, which can be described in the following sense. Suppose P is a predicate (property) and $a_i's$ are individuals and that we from the observations $P(a_1), P(a_2), ..., P(a_n)$ conclude that all individuals have the property P, i.e.,

$$P(a_1), P(a_2), ..., P(a_n) \overset{inductively}{\Rightarrow} (x)P(x)\,[1]$$

This reasoning is going on in a metalanguage in which we can talk about the object and in which we can understand them. From a logical point of view we may express it as that the reasoning is going on in the model.

From the inductive expression it may seem as the numbers of examples are of importance. On the contrary, the characteristic of induction is that there is no *effective* way of deciding the sufficient number n of observations for making the hypothesis (Löfgren, 1982). In fact it is not the knowledge of the number n that is a determining factor. Instead it is the language in which the induction process occurs that is of vital importance. Remark that induction is relativized to languages.

In order to make inductive inferences we must be able to extract regularities from observed sequences of individual cases, to such an extent that it is possible to make a definite description of the phenomena in question. This may be expressed as that induction is the process of making order out of disorder. The shorter the description the more information we get from it in the sense that we have succeeded in extracting more regularity from the surroundings.

Induction is an *information-producing* rule that has no correspondence in formal systems. If we think of inference rules in logic as production rules the case is in fact the opposite:

$$\forall x S(x) \overset{deductively}{\Rightarrow} S(a).$$

That is, all inference rules in logic are *information reducing*. We may also think of it in the following way. Suppose we have a set T of axioms and that φ can be deduced from T. φ is a theorem in the theory T but φ is the result of just one of all possible derivations from T. (In this paper I do not make a distinction between the set of axioms and the theory generated from the axioms.) If all derivations are regarded as branches in a tree, φ is a node in one of those branches. It may be that further derivations can be done from φ, as an inner node in a branch, but the branches above φ is completely impossible to derive from φ. Thus, φ gives less information than T.

Note that in the deduction above (from all x have S to a has S) there is no need to understand the meaning of the symbol S in order to make the deduction. S is just a

[1] (x) means "for all" in the metalanguage.

symbol in a formal language in which the meaning is not part of the deduction. Whatever symbol we choose, the derivation is the same. Of course most symbols in formal languages will be given a meaning but this is not considered part of the language. (Cf. Shoenfield, 1967, p. 4.)

Since induction increases a device's knowledge of the world by creating order, Löfgren (1973) has suggested the following learning hypothesis:

> An object A can *learn* from a surrounding S to the extent that it can extract order (regularities) from S so as to produce a description of S relative to A (A shall be able to make inferences from the description) [...]

This is a general learning hypothesis. Since my intention mainly is to explain model-based anticipatory systems from a computational standpoint I will modify this thesis to one suitable for automata.

All knowledge is about reality or what we perceive existentially in the world around us, man made or not, and since knowledge is a result of learning we may conclude that learning begins with the interaction with the environment. We learn from the surroundings by observations. In doing so, an important component is an already established model of the world. With model I here mean the conception of the world just in the way model is used in logic. In the model *exist* concepts and properties even if they not always are describable. Thus, when we learn something we always do it in some background knowledge. This knowledge is either commonsense knowledge or scientific knowledge. When we learn something it is a universal or existential statement that cannot be explained from the background theory.

Let T be a theory, H a hypothesis and O an observation. Then H increases the knowledge if

$T \nvdash H$ (H cannot be inferred from the theory T).

$T \nvdash O$, and

$T, H \vdash O$ (O can be explained in the extended theory where H is added to T).

It is tacitly understood that T, H and O are consistent.

Thus, H is a description of what is learnt from the surroundings and the shorter the description the more is learnt. That is to say that the shorter the description, the more regularity is perceived in the empirical world and the better is the predictive power of the description.

Now we have a way to measure increase in knowledge and with that also a way to define what should be meant by a learning device. A device which increases its knowledge in the way just described will be called a *genuine learning* device. This leads to the following machine learning hypothesis, which is a modification of Löfgren's hypothesis above.

Machine Learning Hypothesis. Let A be a device with the background knowledge T. Furthermore, let O be an observation (or observation series) that A is able to observe but cannot explain from T. If A can produce a description H from O (relative to itself) such that $T \cup \{H\}$ explains O, then A has learnt H from its surroundings.

With this definition only hypotheses creating devices have a genuine learning capability. It also rules out deductively generated conclusions as learning since everything concluded by deductive devices is already implicit in the premises (axioms). If we again think of a theory as a tree with the axioms in the root nod and all possible derivations branching out from this nod, we cannot claim that it is new knowledge to obtain a nod (theorem) in the derivation tree. In a computational way all trees are the same and what a tree represents is not part of the tree. This is of course an exact equivalence of a derivation in a formal system where a derivation is completely independent of the meaning of the symbols included. As a consequence induction cannot be described in the language of logic since formal systems are deductive systems.

In biology we also talk about learning by *inheritance*, i.e., knowledge genetically transferred from one generation to another. For automata it corresponds to being programmed by the surroundings and consequently is not genuine learning in accord with the machine learning hypothesis. Genuine learning can also be explained as the aim of finding new predicates (concepts) that explain an order of the empirical world.

Since the learning process is a description process it means that there is no absolute learning. We are obliged to stay within a language. It implies that in order to extend our knowledge of the empirical world it is necessary to extend the language with more concepts, which as we have seen cannot be deductively generated. In evolution but also in science this is an adaptation that is continuously ongoing. If certain observations do not fit the existing model it might be so because the model is not a model of the observed data, i.e., we have to change the model or create a completely new model corresponding both to the observed data and to already existing objects. Here again we may refer to Planck's effort to develop his quantum law.

Of course, it may also be the case that the disorder of the observed data is too complex not only to be described in the current background knowledge but also for being describable in any background knowledge. Even in the case we arrive at an object, this can itself be too complex to be described in any theory. As an example of the later we may follow Kleene (1952) who starts with the Turing machine predicate $T(x, y, z)$ meaning that the Turing machine with the code number x will give z as a result when feeding with y on the tape. y and z are coded numbers of the input and output data respectively. We may compare the Turing machine with a computer, where x stands for the program, y for input data and z for output data. Then, the predicate $\neg\exists z T(x, x, z)$ is not describable in any theory. The predicate can be read as "there is no computer that when fed with its own program as input data will produce z as output. This is in fact a generalized form of Gödel's incompleteness theorem, which is explained by Kleene (1952, p. 302, theorem XIII.) as:

There is no correct and complete formal system for the predicate $\neg\exists z T(x,x,z)$.

4 Inductive Inference Machines

Much effort has been spent in mathematics devoted to inductive inference. The reason for calling it inductive inference is due to there being some similarities with the induction discussed above as an information creating method but, as will be shown, this resemblance is very weak. In order to distinguish between the two types of induction I will in this section call the induction discussed above *pure* in contrast to the mathematical inclined induction, which I will call *recursive* of reason that will be obvious.

As for pure inductive inference, the recursive induction starts with some examples (observations). It is a supposition that the underlying set from which these examples are taken is recursive. That is, there is a recursive rule, which generates all values. It is further presupposed that the inference method is computable, which is the same as saying that the method is completely deductive. To sum up, we start with several examples of an unknown recursive function and we are going to find which function by deductive means (Odifreddi, 1999). Let us by an example see how such inferences will work. Suppose we are studying a kind of birds that are either black or white colored but this is unknown from the beginning. Instead the starting point is that we believe that they can have every color so we give all possible colors a number: 0 for white, 1 for black, 2 for red and so on. Now, suppose that we get the following measures:

$$0, 0, 0, 0,$$

that is, the first four observed birds are all white. A computer would immediately suggest the underlying recursive function being $f(n) = 0$. When generating its guess, the computer does not use any background knowledge of biology or any knowledge of the surroundings in which we are making our observations. It always has the same background, namely the language of arithmetic, or to be more precise, an axiom system in which it is possible to do arithmetic (See next section for reference to such a system.)

Suppose now that the next observations are

$$1, 0, 1, 1, 1, 0, 1.$$

It might be that a machine can guess the next value, that is, tell us which recursive function that is likely but it cannot tell us from a conceptual view what seems reasonable, namely that the one half of the population is white and the other black. In order to be able to make such a conclusion the machine must first be able to talk about colors, that is, have the concept of color. It does not help to have a measuring instrument for automatic coding of light frequencies to numbers. Color is not a

physical property but a concept created by the human brain. Nonetheless, color, as an existentially perceived concept, is a reality. Even if a computing machine tells us which recursive function that can generate the initial examples it cannot tell us anything about the concept in question since the computer has no knowledge of color. The numbers used here for colors could be used to codify many other properties. Which interpretation to chose is not part of the numbers. Whatever the code stands for, the computer will guess the same recursive function.

As is also clear from the example above, it is not the observed numbers of initial examples that are crucial for our inductive inference it is our background knowledge that is of importance. Sometimes we generalize from one single example since our background knowledge tells us that we have observed a typical example.

One other supposition in the theory of recursive inductive inference is that the inference is viewed as an infinite process (Odifreddi, 1999). Suppose that M is an inductive inference machine that is capable of describing some unknown recursive function F. If M is fed with more and more examples of the function values, M will generate infinitely many guesses, say $f_1, f_2, \ldots$. If there exists a number N such that f_N is a correct description of F and all further guesses are the same, that is, $f_N = f_{N+1} = f_{N+2} = \ldots$ then we say that M identifies F in the limit. There is no method for a machine to decide when it has reached the point when the guesses continue to be correct. This is called *identification in the limit* (Gold, 1967) and views inductive inference as an infinite process.

Confirming a hypothesis means deciding whether the given series of examples is sufficient to regard the hypothesis as verified, that is, our belief in the hypothesis is so strong that we take it for true. In pure induction there is always a confirming principle in order to decide when our description of the phenomena is sufficient simple to serve as a general description. Furthermore, such a confirmation must with necessity be performed in finite time otherwise we would not arrive at a conclusion. Identification in the limit, as an infinite method, means consequently that the confirmation problem is entirely left out.

Should a finite principle be used in recursive induction it would with necessity be computable otherwise it would not be possible to follow by a computer. It is no accidental circumstance that there is no confirmation principle in machine induction. Löfgren (1982) has shown that no general confirmation principle is recursive.

5 Learning Computers

I will look at computers in their most general form, that of Turing machines. This generality is due to Church-Turing's thesis that can be formulated as that

> *every effectively calculable function is computable on a Turing machine,*
> or equivalently,
> *the Turing machine (as well as any of a number of other models) is a universal model of computation.*

Thus, a Turing machine program is an algorithm for computing a recursive function (Note, every algorithm halts). Such a computation can be carried out for example in the (axiom) system **R** of Raphael Robinson (Smorynski, 1991, p. 338) in which every algorithm can be described. **R** is weaker than for example Peano arithmetic but despite that every recursive function can be represented in it. A basic result in metamathematics is the following connection between computability and formal systems

$$f \text{ recursive} \quad \Leftrightarrow \quad f \text{ is representable in } \mathbf{R}.^{2}$$

Here, f is representable in **R** means that there is a formula φ such that

$$f(x_1, x_2, ..., x_n) = y \;\Rightarrow\; \mathbf{R} \vdash \varphi(\bar{x}_1, \bar{x}_2, ...\bar{x}_n, \bar{y})$$
$$f(x_1, x_2, ..., x_n) \neq y \;\Rightarrow\; \mathbf{R} \vdash \neg\varphi(\bar{x}_1, \bar{x}_2, ...\bar{x}_n, \bar{y})$$

So, every program is in fact a precise description of how to compute a specific function or, in other words, how to deduce a theorem in **R**. However, not all programs are about arithmetic. For example, in logic programming the idea is to define a program as a logical formula and consider a refutation proof a computation. Thus, the computation is a proof in the theory specified by the program. Likewise, database queries are proofs in a theory formulated in the program. A sentence is true in the database (model) if it is deducible from the theory describing the database. In the case of logic programs and databases it is the recursive *inference* rules that are computable and we may regard the program as a theorem prover in the theory represented by the program.

Regardless of the form of the program, it is about something, i.e., the program is a description of a model. In this sense we may consider every program a theory and every execution a proof in the theory. As in every theory, the meaning of the symbols in the program is not part of the program but is added by an outside observer; a program must be specified with its interpretation. This view is well developed by Manna (1974), who defines a program as a pair $P = \langle S, I \rangle$ where S is a flowchart schema (flowchart program) and I an interpretation in model theoretical sense. A flowchart schema can be regarded as a program written in an ordinary programming language. Already in 1947, Goldstine and von Neumann developed the idea of a flowchart program and later it was shown that *every recursive function is flowchart computable* (Odifreddi, 1992). Manna shows as well that every schema can be formulated as a formula in a first-order language and then, again, every computation is a theorem in the specified theory.

The merit of looking at a program as both a description and an interpretation is the emphasize on the non-fragmentability of a language. This has been formulated by Löfgren (1994) as a thesis of complementarity:

---

[2] In fact in a consistent formal system extending **R**.

Language as a complementaristic phenomenon. In general, complementarity refers to *wholistic* situations where fragmentation into parts does *not* succeed. In its complementaristic understanding, the phenomenon of language is a whole of description and interpretation processes, yet a whole which has no such parts expressible within itself. This constitutes a paradigm for complementarity, the *linguistic complementarity*.

This thesis is strongly supported by Tarski, who in his seminal 1936 work showed that no language is allowed to contain its own truth predicate. The predicate "is true" is relativized to whichever language we are speaking of. If we say that a sentence in a language L is true, that is an assertion that belongs to the meta-language, not to the language L.

Manna's view of a program as a linguistic wholeness is not normally what computer scientists and programmers mean by a program. Mostly, if not always, they do not make any clear distinction between a program, as a description, and its corresponding interpretation. Sometimes the code is referred to and sometimes the interpretation of the code. Like mathematicians, programmers mostly reason in the model and not in the formal system. That is why many programmers confuse a program (code) and its interpretation and ascribe properties to programs for which there do not exist formal descriptions. This amalgamation may seem reasonable from a programmer's perspective and is normally not harmful. However, when dealing with basic questions of what computer really can do the distinction is significant. Manna, as well as Löfgren, calls attention to the holistic view of languages: formal as well as natural.

With this insight we can say that an execution of a program is a deduction of a theorem in the theory, which the program constitutes. Expressed differently, a formal system is nothing but a mechanical procedure for producing theorems and Turing machines yield an exact equivalent concept of formal systems. In the light of the reasoning above concerning increasing and decreasing information processes, this can be expressed as that *a computer program is an information reducing process* contrary to induction, which, as we have seen, is an *information creating process*.

That a computation is information reducing can be more easily seen from the view of λ-calculus. If we simplify to one argument, a function f is λ-definable if for some λ-term F

$$F\overline{n} = \overline{f(n)}, \text{ where } \overline{n} \text{ means the numeral of } n.$$

λ-calculus is a theory about computations in the sense of the following relation

$$f \text{ is recursive} \Leftrightarrow f \text{ is } \lambda\text{-definable}$$

Lambda terms denote processes, which means that functions are regarded as rules in order to stress their computational quality. One of the two basic operations is

application, which can be seen as the process of going from input to a rule to output of the same rule, i.e., a computational reduction. By way of an example (Barendregt, 1985, p. 50), take the expression $(\lambda x.x^2+1)3 \rightarrow 10$, where the arrow reads *reduces to*. We can interpret the expression as "10 is the result of computing $(\lambda x.x^2+1)3$". Reduction is not symmetric because there is no way going from 10 to the applicative expression started with. It is in this sense that a computation is information reducing; $(\lambda x.x^2+1)3$ gives more information than just the number 10. Since all λ-definable functions are Turing machine computable, every such computation is information reducing.

Thus, there is no way for a Turing machine to create information, i.e., increase its knowledge. This conclusion could also have been arrived at from a model theoretic reasoning by observing that to be able to create new predicates it is necessary to be able to reason about its own model of the surroundings and in addition be able to change this model. Theory change has the point of departure in the model. The understanding of a theory change must come from outside the theory. It cannot be made from within the theory.

The equivalence between computers (Turing machines) and formal systems has far-reaching implications for machine learning. To exemplify this I will use learning a maze since this is usually regarded as genuine learning. From a computational point of view, learning a maze can be transformed to searching in a tree. Every junction in the maze can be regarded as a nod in a tree and the search can be performed by labeling all visited nodes. Many other examples, where learning is supposed to be involved, are comparable with tree searching, or more general, searching in a graph. As we have seen, all such searches are completely deductive and just a derivation from the premises.

6 Interactive Machines

So far we have exclusively considered conventional Turing machines, which do not interact with their environment. Since modern computers make extensive use of Internet and the interplay with other computers it might be that the *old* Turing machine model no longer is relevant as a model of computation. That this is the case is argued at great length by Wegner (1998).

Already Turing (1939) suggested a machine with an *oracle*, i.e., a machine that is able to halt its computation and request additional information. We may think about such a machine as one that can answer questions about regularities that cannot be answered by any ordinary Turing machine. Following Davis (1958), a Turing machine is a set of quadruples of the following form:

1. $q_i S_j S_k q_l$
2. $q_i S_j R q_l$
3. $q_i S_j L q_l$

where q_i, q_l are states (not necessarily different), S_j, S_k are symbols on the tape and R, L means move *right*, *left* respectively.

In his account, Davis restricts the interrogations permitted by the machine to those of the form

$$\text{is } n \in A \ ?$$

where n is an integer and A is a set of integers, fixed for a given context. As Davis points out, "this limitation is not nearly so restricted as might be supposed". Now, a new instruction

$$4. \quad q_i S_j q_k q_l$$

is introduced with the intention that if the Turing machine contains such a quadruple it should ask whether a certain number on the tape is or is not in the set A.

This provides a Turing machine by a means of communication with the *external world.* When a machine is in a certain state with a certain description on the tape, the machine may be interpreted as inquiring: *If n is a certain number on the tape, is* $n \in A$? The new state of the machine is chosen in accord with whether the answer to the question is *yes* or *no*.

By an A-computation of a Turing machine Z, we refer to a Turing machine that makes use of questions to the set A to finish its computation. Now we can define a function, $f(x)$, to be (partially) A-computable if there is a Turing machine Z, which, with the aid of A, can compute $f(x)$. The oracle is an extra recursive entity, helping the computation of any function recursive in A with its troublesome spots. A call to A can be effectively answered only if A is recursive but in principle, if the oracle is clever enough, any question concerning A can be answered regardless of how the set A is defined.

The definition above can easily be generalized to a set Ψ of completely defined functions instead of a set A of integers. The idea is now modified by assuming that any value of one of the functions in Ψ, if demanded, will thereupon be supplied. A function, which is computable with the aid of Ψ is now said to be Ψ-computable. To simplify matters, I will assume that Ψ consists of just one function, g.

Let R be the class of recursive functions and R^g the class of functions g-computable or *recursive in g*. If g is recursive the two classes are the same while if g is not recursive but notwithstanding *computable* by an oracle, then $\mathrm{R} \subset \mathrm{R}^g$. Now, can a Turing machine replace the function g as an oracle? Yes of course it can, but then the oracle is (partial) recursive so nothing is gained. Hence, there is no point in letting a Turing machine take the place of an oracle since this will not extend computability (Ekdahl, 1999). The computing power is only extended when there is an oracle that can compute *noncomputable* functions.

To increase the learning capability we would have an oracle that can answer questions about the induction function. A human being is such an "oracle" but as we have seen, no Turing machine can take the role of a noncomputable oracle.

7 Anticipatory Behavior

A system that is anticipatory has the ability to foresee the consequences of certain events and act in a way adapted to the purpose of the system. Such behavior can in many cases be performed by means of an embedded algorithmic description that tells the system how to proceed in different situations in order to avoid unwanted consequences. No matter how well suited the description is, such systems are *closed* in the sense that they do not know anything about their surroundings. This does not mean that such a system does not interact with its environment. It just means that the interaction is mechanical and does not involve any understanding of the interaction. It is just the same as when the brake system in a car interacts with the driver: the brake system has neither knowledge of the driver nor of the function of the brake system.

Thus, if the surroundings are changed, the system will not know it because the model, that the description concerns, is not part of the description. sinceIf the surroundings undergo extensive changes, the survival of the insect may depend on new behavior but this cannot be generated from the insect's genotype. The changes must in most cases be caused by mutations, which, from a computer perspective, may be regarded as programmed changes. (In some cases a genotype may lead to different behavior.)

A more developed anticipatory behavior is displayed by systems that in the semantic sense also have the ability to create and reason about a model of the surroundings. Such systems are *open* in the meaning that they perceive their surroundings and may change the model in accord to changes in these surroundings. Systems that possess a model are called model-based contrary to the description-based that only posses a description of their surroundings (Ekdahl, 1997)

Anticipatory behavior involves pre-evaluation in order to avoid unwanted consequences. If the system is description-based, this is a completely syntactic process going on in the language except for the derivation rules, which strictly speaking is a rule outside the language operating on sentences in the language (Shoenfield, 1967, p. 4). I will disregard this distinction and consider the pre-evaluation process going on in the syntactic part of a language. If the outcome of the pre-evaluation does not fit in the surroundings this will not be known to the system. In the case of insects, non-fitting behavior might cause the extinction of the whole species.

For model-based systems the pre-evaluation is performed on a model and in this respect we may speak about model-based behavior. In order to perform the pre-evaluation, such a system makes a description possible to follow if the outcome of the evaluation points to a preferable action. This description process is as we have seen, mostly inductive and accordingly not describable. When a model-based anticipatory

system changes its model it does it on account of an observed change in its surroundings that shows that "Things Ain't What They Used To Be"[3]. The system has observed some order in the surroundings that either overrules a previously believed order or implies a new order not known before, that is, the system has learnt something new from the surroundings. This new order is reached by inductive inference. Since model-based anticipatory systems are the only systems that have inductive capability this gives us yet another way to characterize learning systems, namely as model-based anticipatory. The implication is that computers cannot be learning machines.

8 Conclusion

We cannot go outside the theory in which we are doing our conclusions. All inference rules in logic are chosen to guarantee that we do not say more than we have the right to do. "[...] if one has ten pounds of axioms and a twenty-pound theorem, then that theorem cannot be derived from those axioms" (Chaitin, 1987). It is in that sense that deduction is a process that goes from a higher level of information to a lower.

The whole point with a theory is to be able to deductively predict the consequences of our beliefs, not to create new beliefs. Deductively inferred conclusions make explicit what was implicit in the premises. We cannot go outside a theory and create another theory. For example, it is not possible to create the Euclidean geometry from within Peano arithmetic.

The process of creating new beliefs about the world is a guessing process that has to be supported by observations not fitting in old theories. We arrive at such guesses by inductive inferences, which is the method to generate new ideas. It is necessary to existentially perceive the surroundings, or in other words, to have an existential model of the surroundings in order to have the power of induction. To inductively describe a phenomenon is the same as saying that we have a new existential model of the phenomenon in question. The relation between our beliefs and our model of the surroundings can be described as the following equivalence:

$$\text{New belief} \Leftrightarrow \text{New model.}$$

Thus, if we believe something we did not believe before, then it implies that we have to change model and if we change model it means that there is something that we now believe but did not do before. Since new beliefs are the result of an inductive process, the equivalence implies that the only systems that are capable of maintaining their own model are those with inductive capability. In the classification above of systems it amounts to say that only model-based anticipatory systems are inductive.

Induction is an information creating process with which we continually increase our knowledge of the world. Deduction, on the other hand, is knowledge decreasing in

[3] A tune by Duke Ellington and Billy Strayhorn.

the sense that the result of a deduction contains less information than the premises started from. Thus, inductive capability is a necessary condition for being a learning device. Computers as formal systems are only deductive devices and can consequently not be learning devices. It means that computers cannot create new theories or change their own theory in order to better fit a changed conception of the world. Here we may not confuse the change of model with pure control systems, which can mostly be regarded as error-actuated theories.

References

Barendregt Hendrik Pieter (1985). The Lambda Calculus: Its Syntax and Semantics. North-Holland, second print.

Bridgeman P. (1936). The Nature of Physical Theory. Dover Publication.

Chaitin Gregory J. (1987). Information, Randomness and Incompleteness, Second edition, completely revised, enlarged, reset. World Scientific Series in Computer Science, Vol. 8.

D'espangat Bernard (1999). Conceptual Foundations of Quantum Mechanics, Second Edition, Perseus Books, Reading, Massachusetts.

Davis Martin (1958). Computability and unsolvability. Dover Publications, Inc.

Ekdahl Bertil (1997). Classification of Anticipatory Systems. In Nagib Callaos, Chan Meng Khoong, Eli Cohen (eds.), Proc. 1st World Multiconference on Systemics, Cybernetics and Informatics, Caracas, Venezuela, July 7-11, 1997, Vol. 2, pp. 499-505. Organized by IIIS International Institute of Informatics and Systemics.

Ekdahl Bertil (1999). Interactive computing does not supersede Church's thesis. In Roger Y. Lee (ed.), The Association of Management and the International Association of Management, 17th Annual International Conference, San Diego, California USA, August 6-8, 1999, Proceedings Computer Science, Vol.17, Number 2, Part B, pp. 261-265. Maxmilian Press Publisher.

Gold E.M. (1967). Language identification in the limit. Inf. Control 10, 447-474, 1967.

Goldstine H.H. and von Neumann, J. (1947). Planning and coding of problems for an electronic computing instrument, part II. Report for U.S. Army Ord. Dept., 1947.

Hesse M. (1973). Models of theory-change. In Patrick Suppes, Leon Henkin, Athanase Joja, Gr. C. Moisil (eds.), Proceedings of the fourth international congress for logic, methodology and philosophy of science, Bucharest, 1971: Studies in logic and the foundation of mathematics, vol. 74, pp. 379-391, 1973. North-Holland Publishing Company.

Kleene Stephen C. (1952). Introduction to Metamathematics. North-Holland.

Löfgren Lars (1973). On the formalizability of learning and evolution. In Patrick Suppes, Leon Henkin, Athanase Joja, Gr. C. Moisil (eds.), Proceedings of the fourth international congress for logic, methodology and philosophy of science, Bucharest, 1971: Studies in logic and the foundation of mathematics, vol. 74, pp. 647-658, 1973. North-Holland Publishing Company.

Löfgren Lars (1982). Methodologies and the Induction Problem. in Trappl R., Klir G.

and Pichler F. (eds.), Progress in Cybernetics and Systems Research, vol. VIII, Washington, New York, London: Hemisphere, pp. 15-22, 1982.
Löfgren Lars (1994). General complementarity and the double-prism experiment. In Laurikainen K.V., Montonen C. and Sunnarborg K. (eds.), Symposium on the foundations of modern physics 1994, 70 Years of Matter Waves, Helsinki, Finland, 13-16 June 1994, pp. 155 – 166. Paris, 1994, Éditions Frontiérs.
Manna Zohar (1974). Mathematical Theory of Computation. McGraw-Hill Publishing Company.
Odifreddi Piergiorgio (1992). *Classical recursion theory: The Theory of Functions and Sets of Natural Numbers*. North-Holland, (sec. impr.).
Odifreddi Piergiorgio (1999). Classical recursion theory, Volume II, Elsevier.
Pais Abraham (1991). Niel Bohr's Times, In Physics, Philosophy, and Politics. Clarendon Press.
Papadimitriou Christos H. (1994). Computational Complexity. Addison-Wesley Publishing Company, Inc.
Shoenfield Joseph R. (1967). Mathematical Logic. Addison-Wesley.
Smorynski Craig (1991). Logical Number Theory I. Springer-Verlag.
Tarski Alfred (1936). Der Wahrheitsbegriff in den formalisierten Sprache. Studia Philos., 1, 261-405, 1936. (English translation 1956 in Logic, Semantics, Metamathematics, Oxford, (Second Edition, second printing 1990, J Corcoran (ed.), Hacklett)
Turing Allan (1939). Systems of logic based on ordinals. Proceedings of the London Mathematical Society, vol. 45, pp. 161-228, 1939.
Wang Hao (1987). Reflections on Kurt Gödel. A Bradford Book, The MIT Press.
Wegner Peter (1998). Interactive foundations of computing. Theoretical Computer Science 192, pp. 315-351, 1998.