back next

Europhysics News (2002) Vol. 33 No. 2

Will machines start to think like humans?

Artificial versus natural Intelligence

G. Binnig*, M. Baatz**, J. Klenk**, and G. Schmidt**
****IBM Research Division, Zurich Research Laboratory, CH-8803 Rueschlikon, Switzerland,***
*****Definiens AG, Trappentreustr. 7, D-8803 Munich***

***T**hinking is a complex procedure, which is necessary in order to deal with a complex world. Machines that are able to help us handle this complexity can be regarded as intelligent tools that support our thinking capabilities. The need for such intelligent tools will grow as new forms of complexity evolve, for example those of our increasingly globally networked information society.

In order for machines to be capable of intelligently predigesting information, they should perform in a way similar to the way humans think. First, this is a necessary prerequisite in order that people can communicate naturally with these machines. Second, humans have developed sophisticated methods for dealing with the world's complexity, and it is worthwhile to adapt to some of them. Such "natural thinking machines" (NTM) with their additional conventional mathematical skills can leverage our intellectual capabilities.

It appears that NTMs require new software approaches as well as novel hardware solutions, which might possibly be based on nanotechnology. However, in this paper we concentrate on the principles of "natural computing" (performed by NTMs), independent of their realization in actual soft- or hardware.

The capabilities of today's machines are still far from humans' capabilities. However, natural computing has not yet reached its limits. Existing approaches combined with new concepts and ideas promise significant progress within the next few years. The vision of natural computing as a new concept presented in this paper is based on our understanding of the principles of human thinking combined with the functional principles of biological cell systems.

Established approaches
The field of artificial intelligence knows several different approaches to model natural thinking. They include semantic networks [1], Bayesian networks [2], and cellular automata [3], but the most prominent examples are neural networks [4] and expert systems [5]. At the same time, modern programming tools frequently have a touch of natural computing [6] to enable programmers to generate code for increasingly complex problems. We believe that the basic principles of the established approaches discussed below will be fundamental elements of future natural computing. In the section "Triple-S network" we will show how they can be combined into one tool and propose new aspects to be added.

Expert systems (ESs) express the logic of human thinking in tree-like decision structures

Expert systems (ESs) express the logic of human thinking in tree-like decision structures. This yields satisfactory results for medium-sized systems. For increasingly complex structures, however, this approach reaches its limits. The net-like aspects that bring associations, abstractions, and uncertainties into play-possibly the most important feature of humans' talent to deal with complexity-is not simulated adequately.

Neural networks (NNs) obviously concentrate on the network aspect. They represent a very basic form of natural computing. In essence, NNs simulate the fundamental functions of the nano- and microstructure of the brain, its neurons, and their interconnections. Input signals are usually connected to this network and propagate in a certain way throughout the network. How the signals propagate can be controlled by weighting the connections between the nodes. These weights can be adjusted automatically by means of a training procedure.

Tasks such as recognizing faces, which might not be easily dealt with in logical terms, are the strength of NNs. Extremely complex tasks, however, would require that large amounts of interwoven complex concepts, like those present in our brains, can evolve automatically. This would be equivalent to reinventing the evolution of our mind from scratch.

Semantic networks (SNs) and (semantic) ***Bayesian networks*** (BNs) have certain similarities with NNs, as they also employ probabilistic methods. Here, however, the nodes and, in some cases, the links are connected to individual semantic expressions, i.e., they have names or expressions associated with them. Therefore these networks have the advantage of being accessible locally in their internal structure to humans. A disadvantage compared to NNs is that the network structure has to be created manually, but this structure can later be checked for its plausibility and modified if necessary because the nodes and links carry semantic, human-understandable meaning. Limitations arise when these networks become large and complex, in which case it becomes increasingly difficult to oversee the overall performance of the network.

For BNs the structure of the network is built manually, and the weights can be trained. Unlike in NNs, the weights on the connections to a node are not treated as being independent from one another. Technically this is achieved by means of matrices of weights on the nodes instead of weights on the links. As a result the number of weights might become extremely large.

In SNs the links as well as the nodes can carry meaning. For instance, simple facts such as "legs are part of a person" and "man is more specific than person" can be expressed using hierarchical links such as "is part of" and "is more specific than".

SNs and BNs generally use global algorithms that operate on the networks to calculate excitation values of the nodes. Some SNs contain "procedural attachments" at the nodes, which are triggered when their nodes' excitations exceed a certain threshold.

Programming languages. The most common way to introduce artificial intelligence is to write a conventional computer program in a common language such as C++ or Java to perform, for example, a cognition process. Typical examples are search engines for texts based on statistical methods or filters in images that extract certain objects. The task here is to find the smartest algorithms and combine them in the smartest way. This is done manually by programmers, and it is up to them to make the program appear intelligent. If in an image of a computer chip a broken line is found automatically, or if a chess-playing computer beats the world champion, there is certainly intelligence behind this process. It is virtually impossible to write an error-free program of reasonable complexity, and it is equally impossible to oversee what such programs do. That is why programs have to be debugged, their performance tested, and revised again and again. If programmers had to write these programs in machine language, the task would be nearly impossible. Modern computer languages enable programmers to write code on a higher level, i.e., more like the way they think and less like how the processor works. Hence modern programming languages are tools that reflect human thinking to a certain extent, and therefore constitute a kind of natural computing.

If a person could teach a computer in a natural way how to perform a certain task, one certainly would call it a natural thinking machine. If one looks into the history of programming languages, one finds that they have evolved in this direction. Object-oriented and other modern languages indeed use many natural concepts such as classes and inheritance.

Cellular automata are instruments to compute complex situations. The idea behind cellular automata or cellular machines (CMs) is quite natural: neighboring objects influence each other. A large number of so-called cells in a regular geometrical arrangement are used. The cells usually have discrete states that are influenced by their relationships with their neighbors. CMs can be regarded as primitive networks, but their dynamics can be extremely complex. In principle (disregarding performance) any kind of computation can be performed with them.

Triple-S network, fractal machine

If one combined all of the above-mentioned methods or at least certain aspects of them into one technology, one could conceivably accumulate their individual strengths to produce a more powerful machine. Figure 1 proposes such a method in the form of a particular knowledge network. As discussed below it is a *self-organizing, semantic, self-similar network*, or a triple-SN for short. A triple-SN is essentially a kind of hierarchical world knowledge network containing knowledge about objects, their properties, and their relations, as well as processing knowledge about what to do when certain kinds of objects are present in the real world. By "real world" we mean the varying input that interacts with the triple-SN. This input could be an image, a text, or any complex structure.

There are nodes and links that carry semantic meaning (similar to SNs) as well as procedural attachments, which are shown as Jani (singular: Janus: a god with two faces). Some links represent ES-like logic ("and", "or", and other more complex functions). Links and nodes carry weights that can be trained (similar to NNs). Links can be linked to other links, which allows (in addition to other aspects) dependencies to be introduced among them (similar to BNs).

The most unusual feature of our triple-SN is the combination of a hierarchical structure and the large number of procedural attachments [7]. This constitutes a form of generalized CM: the state (activation) of a Janus depends on the state (excitation) of the node to which it is attached, whose state depends in turn on the states of the neighboring nodes, Jani, and links. In the triple-SN, nodes and connections are grouped into subnetworks. If these subnetworks are also regarded as nodes with associated states, we end up with a hierarchical, generalized CM with similarities across the hierarchies, which we call a "fractal machine".

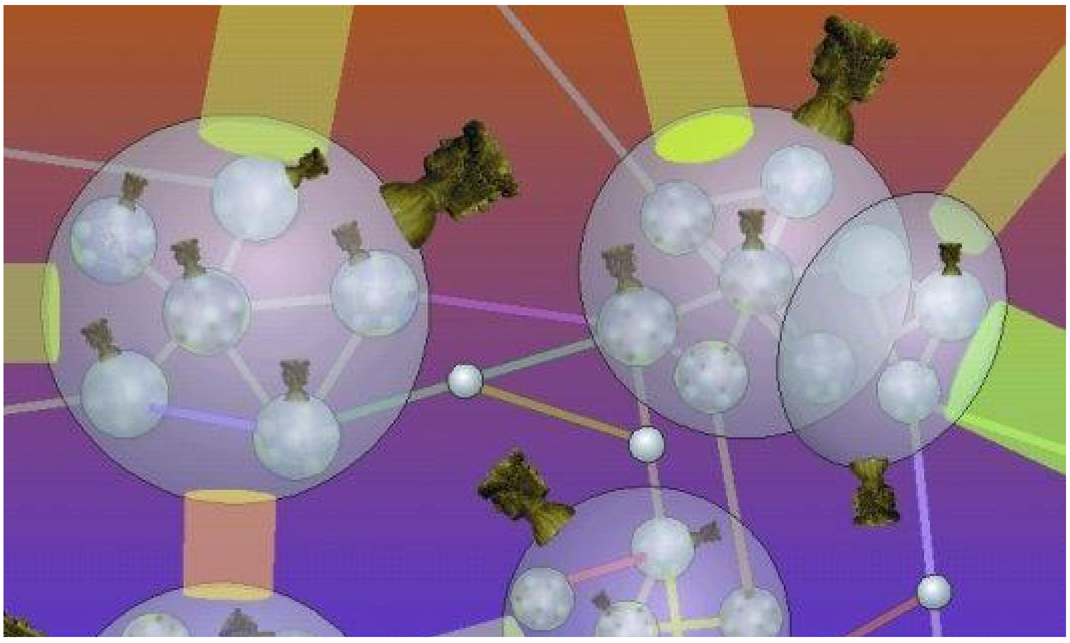
When such a network interacts with complex inputs, inheritance (in the sense used by modern programming languages) comes into play: input objects (acting as instances) inherit Jani from nodes or links (acting as classes) to which they fit best. This inheritance can also be regarded as the activation of procedural attachments. The objects in the input change their states stepwise according to the influence of their neighborhood (similar to CMs). Some of the Jani represent classification procedures, which compare classes and instances with each other, whereas other Jani network or group the input objects (segmentation). Thus, in a stepwise procedure of alternating classification and segmentation, the activated Jani transform the initially unstructured input into a hierarchical network. The structure of the input network therefore becomes increasingly similar to the triple-SN itself. For example, at the onset, an input image might consist only of various pixels. In the course of the procedure, a hierarchical, networked structure of the image evolves in steps to produce shapes such as houses with neighboring objects such as streets, which ultimately evolve into a city. Attributes are attached to the objects and their relations. In the triple-SN this structure is also expressed as one of several possible structures.

With this approach, more objects and relations are created than necessary. Some of them are discarded. Others, which are relevant of themselves or useful for creating objects on a different hierarchical level, are kept. The creation of objects and their relations on and across different hierarchical levels is equivalent to transforming information into knowledge. This includes context as it is represented by the local network neighborhood of a particular object.

The automatically alternating procedures of classification and segmentation of the input objects is a new aspect, which we call self-organizing or affective computing. Two other aspects that have previously been overlooked are generic computing and self-similar (or fractal) computing. All three mechanisms represent important mechanisms of nature and will be discussed further.

Affective (self-organizing) computing. Let us assume for a moment that a living organism can be regarded as a kind of intelligent "computer" that automatically responds to changes in its environment by starting certain "programs". Our brains are clearly capable of doing so. In the course of our lifetime we learn a huge amount of different behaviors and strategies. They are all coupled to certain classes of situations. Only if we assign a situation to a certain class (classification) will we begin to apply the strategy that fits the situation (affect). All other strategies are dormant. It is important to note that, apart from strategies of action, different classification methods also represent different strategies, which are usually classification-driven as well. For example, with the help of classification our mind is able to concentrate on certain aspects of a given thing and to subject it to a more detailed classification.

Something very similar is true for a primitive living organism such as a cell system. One could regard the code of the DNA-strand present in each cell as programs that represent strategies. In this sense the genes (the relevant segments on the DNA strands) represent individual programs, and the exons (the smaller segments of the genes) represent subprograms.



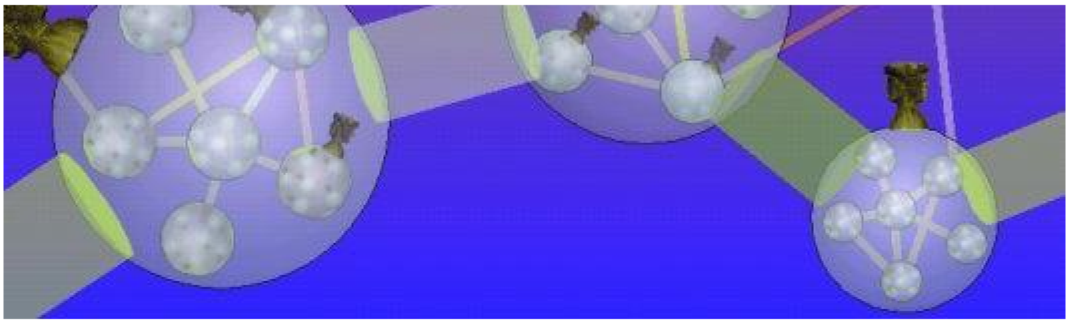


Fig 1 **Triple-S network: self-organizing, semantic, self-similar network; a kind of hierarchical world knowledge network. It contains knowledge of objects, their properties, and their relations, as well as processing knowledge on how to handle certain classes of objects and relations when their instances are present in a complex input. The hierarchical structure of objects containing networks of smaller objects and locally attached algorithms is visualized by Jani (head with two faces). The triple-S network uses activated Jani to structure and network complex inputs as well as to classify the evolving objects and their relations. Jani are activated when the attached objects and relations become excited beyond a certain threshold. Their particular operation is influenced by the local structure of the network. The networking and structuring of the input transforms information into knowledge and, to a certain extent, constitutes an automatic "understanding".**

Activating these individual programs means reading the code of a gene and transcribing it ultimately into a protein. In this sense the protein represents the activated gene and actually performs certain tasks. Similar to the affective behavior mentioned above, at a given moment or at a specific location most of the strategies (in this example, the genes) are not activated. The pertinent genes are only transcribed if required by the location or the momentary situation of the cell. It is a kind of classification process that triggers the activation in a way similar to affective behavior. Here again some of the activations triggered might represent detailed classification procedures.

There are a large number of steps involved from reading code to producing RNA-strands to generating protein code to folding the protein a specific way. Many things can "go wrong" in producing the final protein. For example, if the molecules and proteins needed during the various steps are missing, the protein cannot be formed properly. There is doubtlessly intelligence behind the processes in the cell, for we can assume that if something goes wrong in one of the steps, the final protein might in fact be discarded. Then, in a sense, the total process from transcription to folding of the protein can be regarded as a classification process. However, there is more to it than an isolated classification process. As several other types of proteins are involved in the production of a particular protein, the system constitutes a complex network with a generalized cellular machine character: the state of a gene (its activation) depends on the state (activation) of other genes.

A DNA strand looks like a one-dimensional piece of information. The same is true for a text, for example, or a thesaurus. In fact all these strings represent network because the objects therein are related to one another. If the meaning of a word or an image object (its state or classification) depends on the meaning of other words or objects in its neighborhood, they form a network with this CM character.

Generic or fractal computing. The proteins in a cell are part of certain processes called pathways. It is interesting to note that several proteins are usually part of a pathway and that one kind of protein can be part of many different pathways. The latter circumstance would be equivalent to a computer subprogram that constitutes an essential part of very different programs, such as an image-processing or a translation program. For primitive subprograms such as the command "copy", this is already the case today. This command can be applied to objects in images, in texts, or to entire documents. In general, however, subprograms cannot objects in images, in texts, or to entire documents. In general, however, subprograms cannot be used in any other program.

If the function of proteins were not based on such a generic principle, then a complete, new set of proteins would be required for each pathway and the genetic code would be orders of magnitude longer. The same is true for a computer program with subprograms that can be used repeatedly for very different tasks: the total length of the source code can be reduced dramatically. The variety of what the programs can do results from the variety of ways in which these subprograms can be combined. We suspect that a large portion of evolutionary effort went into formulating genes in a generic way, and this portion is possibly even larger than the portion that went into finding functional genes. How could such a short code in our DNA strands describe a system as complex as a human being? The answer is generic processing.

For computer programs this could apply as well. In the past, programs have been formulated to perform certain tasks. In the future they might be formulated in a generic way, so that their components can be used for other kinds of programs as well. If (sub)programs are formulated in such a way that they can be combined with other programs, the complexity of a new task can be handled not by writing a completely new program, but simply by creating a new combination of existing ones. The richness of the new program results from the richness of all possible combinations.

We believe that this property will be essential to handle complex computing tasks of the future. A future search engine might have to analyze the text as well as the images in documents, or other kinds of data. If the task is simple and both analyses can be done independently, the problem could be solved in a conventional way. If certain parts of the text relate to certain parts of the image, which is usually the case, the analysis is entangled, and a detailed cooperation of various programs is needed. This cooperation is much easier if the programs have similar structures, just like cooperation between people is easier if they speak a common language. The simplification of the interfaces is what makes the difference. If people speak the same language, they need no translation.

The simple command "copy" in an operating system of a computer is useful across many hierarchies. A character, a word, an image, a text, a document, or a folder can all be copied. In a cell, the generic structure is present over many hierarchies. Proteins are part of a single pathway, but they are also part of the entire functionality of the organism. In our brains, the concept of similarity can be applied to practically everything and across all hierarchies. Two different objects in an image might be alike, two words might be synonyms, two texts or two images might have similar meanings and so on. We call a system with generic components that are used on many different hierarchical levels a "fractal" or "self-similar" system.

We expect that it will soon become possible to extract meaning from complex inputs such as documents, regardless of the format of the content. Computers will "understand" multimedia documents to a still somewhat primitive extent, but well enough to constitute a useful intelligent assistant for humans.

Acknowledgment
We thank Maria Athelogou, Michael Alvers, Peter Bloechl, Florian Eckhardt, Robert Kiendl, and Arno Schaepe for very stimulating and helpful discussions.

Literature:

[1] R.J. Brachman. "What's in a concept: Structural foundations for semantic networks," International Journal of Man-Machine Studies **9**, pp. 127-152 (1977); W.A. Woods. "What's in a link: foundations for semantic networks". Reprinted in "Readings in Knowledge Representation," ed. R.J. Brachman, H.J. Levesque.

[2] P. Spirtes, C. Glymour, and R. Scheines, "Causation, Prediction, and Search", Springer, New York, 1993; F.V. Jensen, "An Introduction to Bayesian Networks," Springer, New York, 1996.

[3] J. von Neumann, "Theory of Self-Reproducing Automata," University of Illinois Press, Illinois, 1966. Edited and completed by A.W. Burks; S. Wolfram. "Cellular Automata and Complexity: Collected Papers," Addison-Wesley, 1994.

[4] B. Müller, J. Reinhart, M.T. Strickland, "Neural Networks: An Introduction," Springer, 1995.

[5] James P. Ignizio, "Introduction to Expert Systems: The Development and Implementation of Rule-Based Expert Systems," McGraw-Hill, 1991.

[6] Martin Abadi and Luca Cardelli, "A Theory of Objects," Springer, 1996; David Ungar, Craig Chambers, Bay-wei Chang, Urs Hölzle, "Organizing Programs Without Classes," Lisp and Symbolic Computation 4,3,1991; Kluwer Academic.

[7] J. Klenk, G. Binnig, E. Bergan, "Modeling Knowledge and Reasoning Using Locally Active Elements in Semantic Networks," Proceedings of ES 2001; J. Klenk, G. Binnig, G. Schmidt, "Handling Complexity with Self-Organizing Fractal Semantic Networks," Emergence, 2(4), pp. 151-162, Lawrence Erlbaum Associates, Inc. (2000).

Gerd K. Binnig received numerous awards including the Nobel Prize in Physics (1986), for the development of the scanning tunneling microscope, which he invented together with Heinrich Rohrer. The scanning tunneling microscope and the atomic force microscope which he invented later made it possible to image and study structures and processes on the atomic scale. These instruments, which serve as tools for investigations of phenomena of the smallest dimensions, play a key role in nanoscience and nanotechnology.

Binnig's current research is concentrated on micro- and nanosystem techniques, specifically on the development of a novel nanomechanical data storage system, and on the theory of "Fractal Darwinism", which he developed to describe complex systems.

back next