



# Controlling Artificial Intelligence: An NST Perspective

Normalized Systems White Paper

---

May 22, 2026 | Herwig Mannaert



## Contents

---

|          |                                                  |           |
|----------|--------------------------------------------------|-----------|
| <b>1</b> | <b>Introduction</b>                              | <b>2</b>  |
| <b>2</b> | <b>Origins of Artificial Intelligence</b>        | <b>2</b>  |
| 2.1      | Pattern Recognition and ANNs . . . . .           | 3         |
| 2.2      | ANNs and Large Language Models . . . . .         | 4         |
| <b>3</b> | <b>Entropy: Uncertainty vs. Structure</b>        | <b>6</b>  |
| 3.1      | From Spreading Out to Uncertainty . . . . .      | 6         |
| 3.2      | Controlling Uncertainty with Structure . . . . . | 6         |
| <b>4</b> | <b>Modular and Hierarchical Systems</b>          | <b>7</b>  |
| 4.1      | The Architecture of Complexity . . . . .         | 7         |
| 4.2      | Modular Structures in Software . . . . .         | 8         |
| <b>5</b> | <b>Life, Evolution, and Modularity</b>           | <b>9</b>  |
| 5.1      | Modular Building Blocks in Life . . . . .        | 9         |
| 5.2      | On Evolution and Normalization . . . . .         | 10        |
| <b>6</b> | <b>Conclusions</b>                               | <b>11</b> |

# 1 Introduction

---

Normalized Systems Theory (NST) is a theory on the behavior of modular structures under change. The theory derives principles or necessary conditions for such structures to avoid that they become coercive and hamper evolution. On the other hand, NST proposes detailed structured architectures to provide much higher levels of evolvability than currently available. To enable the large-scale application of NST in software engineering, code generators or so-called expanders have been created by the university spin-off company NSX bv to generate such evolvable structures for software applications.

Today, the automated generation of software by Artificial Intelligence (AI) tools, more specifically Large Language Models (LLM's), is rapidly spreading in the software engineering industry. In this White Paper, we explain that these code generation techniques are not competitive but rather complementary with respect to NST code generation techniques. More specifically, we will explain that there is a crucial need for the AI code generating models to provide additional hierarchical structures to control the near infinite number of possible combinations in creating software programs. On the other hand, while NST expanders already provide the necessary hierarchical structures, the development of the additional custom code extensions, currently a manual and therefore time-consuming activity, will greatly benefit from the automated coding capabilities of the current AI models.

In Section 2, we give a concise overview of the origins of current AI models in pattern recognition, as this is quite relevant to understand the behavior of these models. Section 3 starts from the fundamental concept of entropy to explain how the amount of possible combinations from basic primitives, such as sensory input values and software coding lines, can quickly become nearly infinite if no hierarchical structure is being imposed. Section 4 elaborates how both man-made artifacts and biological systems apply such hierarchical structures to cope with these nearly infinite possible combinations. In Section 5, we look at the evidence that such hierarchical modular structures are intimately connected to life and evolutionary systems, as they are not only essential to cope with environmental changes, but also arise naturally from such variations. Finally, we present some conclusions in Section 6.

## 2 Origins of Artificial Intelligence

---

Both the notion and the term **Artificial Intelligence** are nearly as old as computers and computer programming. The term was first coined during the summer of 1956 at the *Dartmouth Summer Research Project on Artificial Intelligence*. The notion lead almost immediately to some inflated expectations, as Herbert Simon and Alan Newell stated for instance in 1957 that “*within ten years a digital computer will be the world's chess champion*”, and “*within ten years a digital computer will discover and prove an important new mathematical theorem*”.

While other streams existed, such as *fuzzy logic* and *genetic algorithms*, two main streams emerged in the subsequent decades. Grounded in *predicate or first-order logic*, the discipline of **Knowledge-Based Systems** emerged, widely known for its *Expert Systems*. Developed by so-called knowledge engineers in declarative or knowledge-based languages such as Lisp and Prolog, these systems aimed to assist and/or replace highly skilled experts for specific tasks, for instance in the area of medical diagnosis. After some initial inflated expectations in the 1980's, the domain matured in some less ambitious forms like *Business Rules*, and seems to be regaining attention recently under the name *Symbolic AI*.

Parallel to these knowledge-based systems, a second main approach toward the creation of AI systems emerged based on models of the elementary buildings blocks of the human brain. These **Artificial Neural Networks (ANNs)** mimicked the structure of *Biological Neural Networks*, and were mainly developed and used in the area of *pattern recognition*. Similar to the knowledge-based systems, the domain of ANNs

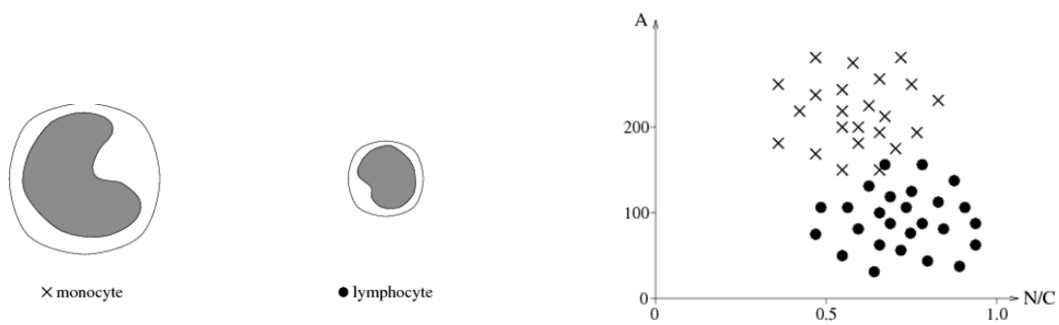


Figure 1: Pattern recognition example: two types of blood cells with a basic feature vector.

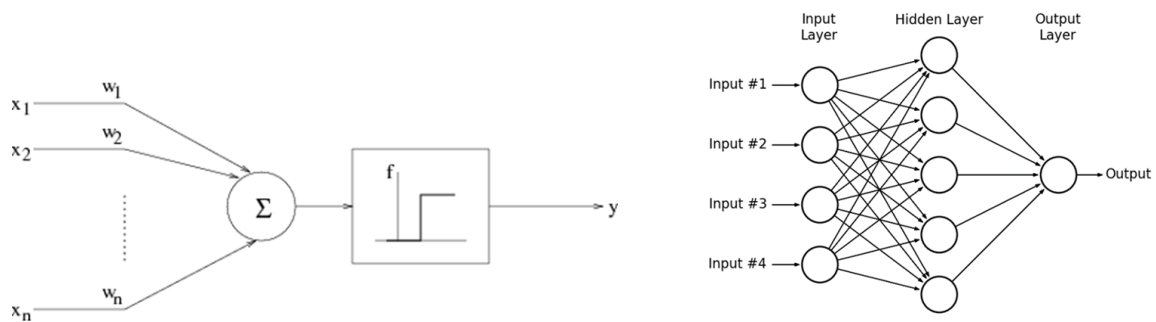


Figure 2: Representation of a McCulloch-Pitts neuron (left) and a multi-layer perceptron (right).

had soe inflated expectations in the 1990's, befor ematuring in some less ambitious forms like *Business Intelligence* and *Data Mining*. However, ANNs made a huge comeback when *ChatGPT* and subsequent *Large Language Models (LLMs)* stormed the world in November 2022.

## 2.1 Pattern Recognition and ANNs

Pattern recognition originated in the area of *signal processing* as a means to automatically analyze and interpret signals such as images, sounds, and electrical signals like an electrocardiogram (ECG). In general, patterns are recognized through the classification of so-called *feature vectors*. A very basic example is represented in Figure 1. Images of two types of blood cells, monocytes and lymphocytes, are being recognized through a feature vector with two features or dimensions: the total cell area  $A$  and the ratio  $N/C$  of the area of the nucleus with respect to the total cell area. Through the use of the feature vector, the pattern recognition is reduced to a simple *classification problem*, i.e., to which of the two classes does a feature vector belong. This classification problem is pretty straightforward and can even be solved analytically through the use of probability distributions and Bayesian classification. Nevertheless, such classifications become complicated in practice for two main reasons.

The first issue is that both the number of features and the number of classes can become quite large, precluding an analytical solution. Often, the number of classes cannot even be known upfront. This led to the development of so-called *clustering algorithms* to not only classify the feature vectors, but to automatically find the (number of) clusters, i.e. the classes, as well. It is within this context of automatic clustering and classification that ANNs emerged as a viable technique. These ANNs emulated the processing of BNNs by combining three basic models, schematically represented in Figure 2.

1. The **McCulloch-Pitts neuron** models a single neuron as a processing unit that sums a number of inputs  $x_i$ , each with a corresponding weight  $w_i$ , followed by a non-linear threshold, such as a step or sigmoid function [1].

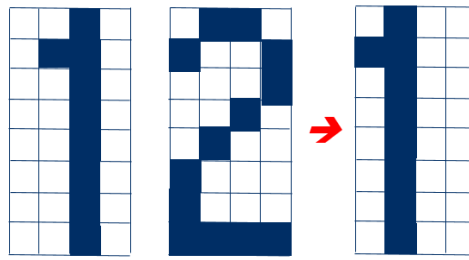


Figure 3: The direct sensory input of some numbers on low resolution images.

2. **Hebb's Rule** models the learning and adaptation of the BNNs through the adaptation of the weights  $w_i$  [2]. Every time the neuron *fires*, i.e., its output is activated through the step function, the weights of all inputs that have contributed to this output are increased.
3. This adaptation of the weights is generalized for so-called *Multi-Layer Perceptrons (MLPs)*, i.e., ANNs consisting of multiple layers, through the concept of **backpropagation**, as proposed by *Rumelhart-Hinton-Williams* [3]. When the weight of an input node that has contributed to the firing of the output is increased, the weights of the neurons that have contributed to the firing of that input node are increased as well. This mechanism can and is propagated backwards through multiple layers.

Such *Multi-Layer Perceptrons* have been used for decades in pattern recognition, where the feature vectors are passed as inputs  $[x_i]$ , and the outputs represent the classes. During the training phase, the appropriate classification is learned through the adaptation of the weights.

The second — and far more serious — issue however has always been that in practice to evaluate or compute the feature vectors. Indeed, the segmentation of objects in images, a prerequisite to compute features like area, is far from trivial. And if segmentation and feature computation is erroneous, it cannot be restored as the system has moved on from the original data to the feature vectors. Therefore, pattern recognition with ANNs was heavily dependent on the so-called **preprocessing**, i.e., the computation of features. Or to quote the famous Carver Mead speaking in 1990 to the ANN researchers: “*I want to perform enough preprocessing to give you something worth learning*”.

Using the sensory input data directly as input to the ANNs is not a straightforward solution either. Consider for instance the basic representations of numbers in low-resolution binary images Figure 3. By merely shifting the number 1 by one pixel, it has more pixels in common with, i.e., is more similar to, the number 2 than the other number 1. As we can have for instance various translations, rotations, and scalings, it is obvious that we cannot simply use direct sensory inputs without preprocessing for classification purposes. Even when we would have large enough ANNs to cope with non-binary images having realistic dimensions. Moreover, anecdotal evidence exists that such fallacies related to direct sensory inputs without *proper preprocessing* still exist today in large ANNs. For instance, an ANN performing extremely well in distinguishing between wolves and huskies, was found to mainly use the presence of snow in the image, resulting in a false classification when a wolf was portrayed in the snow. Or an ANN diagnosing skin cancer in an accurate way, was found to have learned that judgement based on the presence of a measuring device in the image.

## 2.2 ANNs and Large Language Models

After the 1990s, the growing computing power enabled the use of larger ANNs, with a larger amount of inputs and outputs, and ever more internal layers and nodes. Such a network, considerably larger than the multi-layer perceptron of Figure 2, is schematically represented in Figure 4. While these larger networks were still not powerful enough to address realistic computer vision tasks, they proved to be

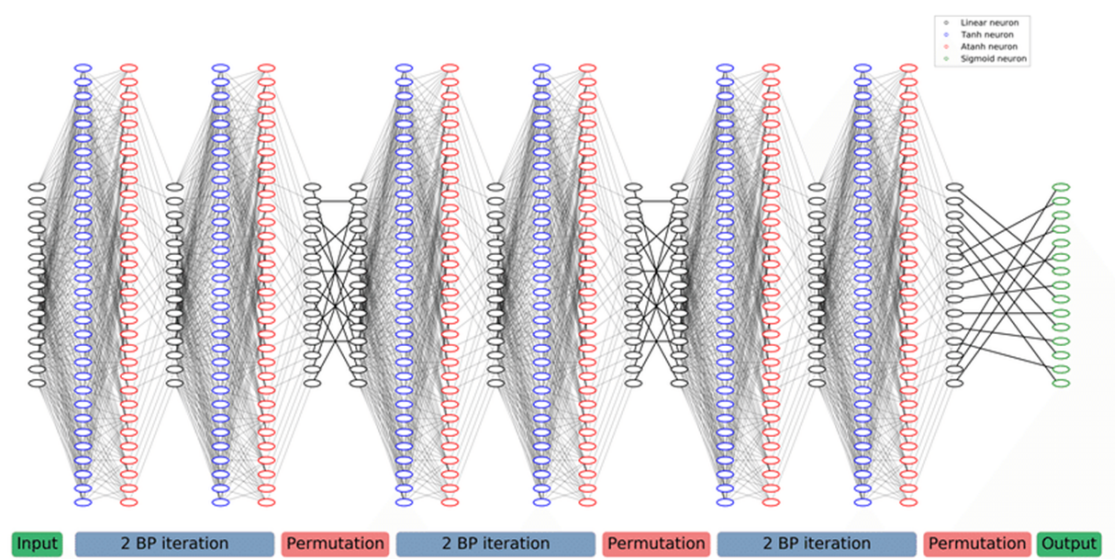


Figure 4: Representation of an ANN with significantly larger amount of layers and inputs.

quite powerful when the the input data and the possible classifications exhibited lower dimensions. Typical examples included *Business Intelligence* and *Data Mining* applications.

A completely new domain of ANN applications opened up when researchers started to use high-dimensional feature vectors to store words and sentences as a list of numbers. These feature vectors, called **embeddings** were made to *carry the semantic essence to provide context*, i.e., similar things end up close in that space, and the Euclidean distance is a measure for similarity and semantic relations. The large ANNs used to represent and generate natural language, have become known as **Large Language Models (LLMs)**.

Considered to be the major breakthrough in LLM technology, the seminal 2017 paper “*Attention Is All You Need*” by Ashish Vaswani et al. [4] introduced the *Transformer architecture* for sequence transduction tasks such as machine translation. Its core insight and essential contribution was proving that attention mechanisms alone are sufficient to model dependencies in sequences. By leveraging stacked encoder-decoder layers built purely around self-attention the model became far more parallelizable enabling to achieve benchmarks at a fraction of the training cost, and laying the foundation for essentially all modern large language models.

Notwithstanding their advanced architectures, these **Generative Pre-Trained Transformers** remain in essence a *next-word prediction machine, pre-trained and fine-tuned to predict the next word or token*. Similar to the issues that those networks face in image recognition, as described in Section 2.1, the processing of large amounts of direct language inputs, without proper preprocessing, poses serious pitfalls. And similar to the fallacies encountered in image interpretation, the tenacious occurrence of hallucinations in modern LLMs indicate that such *proper preprocessing is not straightforward*.

On top of these conceptual issues, ANNs in general and LLMs in particular face some serious challenges related to their energy consumption. While a human brain operates at around 20 Watt, a single computer rack in an AI data center consumes north of 20 kWatt. As these data centers consist of many corridors full of such computer racks, their energy consumption is in the order of MWatts or even GWatt. It is important to realize that this energy issue is not coincidental, but rather fundamental. Indeed, the biological neuronal networks, are very much aligned to their technology, with cell growth supporting massive amounts of three-dimensional connections, and electrochemical transmitters enabling signal processing with extreme low energy consumption. Using the processing model of BNNs with a totally

different technology is bound to lead to excessive energy consumption.

### 3 Entropy: Uncertainty vs. Structure

---

We argue that the need for proper preprocessing in ANNs is related to the sheer infinite amount of possible combinations in both direct sensory and language inputs. To understand and describe the relationship between the number of possible combinations and engineering complexity, the concept of entropy has been introduced in engineering sciences.

#### 3.1 From Spreading Out to Uncertainty

Entropy was first introduced in thermodynamics based on the finding that the integral of the heat exchange standardized by the temperature was independent of the path. This means that heat normalized by temperature is a so-called *state function*, i.e., a property solely dependent on the state of a system and not on the history. That property was called **entropy**, and is physically related to *the degree to which the heat, or more general the energy, is spread out across particles*. The concept of entropy is perhaps best known through the *Second Law of Thermodynamics*, stating that *The entropy in a closed system always increases and evolves naturally to an equilibrium state of maximal entropy*, expressing nature's irreversible tendency to spread out.

Boltzmann proved that thermodynamic entropy is related to the number  $W$ , *the amount of possible microstates corresponding to a given macrostate*. This *number of realization possibilities*, i.e., the total number of all possible states of all individual particles that may correspond to the same observable macroscopic state, is the basis for the definition of entropy in statistical mechanics, and unlocks an even deeper meaning of entropy. Representing the number of realization possibilities, entropy becomes a measure for both the *uncertainty of a state*, i.e., the number of detailed internal microstates that may be underlying the observed macrostate, and the *probability of the state*, i.e., the number of possible microstates that may have led to this observed macrostate. In other words, in a closed system, the most probable state is the state that exhibits the highest uncertainty.

As the second law of thermodynamics concerns closed systems, entropy increase is not unavoidable in an open system, where additional energy or effort can be applied to mitigate the increase of entropy. Nevertheless, nature's natural tendency to spread, to interact, and to defragment is inconvenient for engineers desiring to order and fragment artifacts and constructions. Moreover, the tendency to mix into indistinguishability is highly inconvenient for diagnostic purposes. For instance, medical doctors want as little uncertainty as possible to diagnose the internal state based on macroscopic observations. And the same argument holds for engineers who want to identify the root cause of a rocket failure based on the observed macroscopic situation, and for software engineers who want to find the precise location of a software bug based on the observed erroneous behavior of a software program.

#### 3.2 Controlling Uncertainty with Structure

An effective way to limit the uncertainty or indistinguishability associated with a specific macroscopic state is *to introduce structure*. For instance, engineers will introduce many sensors in rocket systems to pinpoint where exactly things started to go wrong. And software engineers will integrate many log statements in software to be able to locate in a precise way bugs in source code. This fundamental relationship between uncertainty and structure is schematically represented in Figure 5. In the top row, we see gas molecules after being confined in the upper left corner of a container (image on the left), diffusing all over the container when the structural partitioning is removed. As the gas molecules spread out, the number of possible microstates, and therefore the entropy, increases. In the bottom row, we have a similar setup with 100 coins, equally divided over 50 coins heads up and 50 tails. When all 100

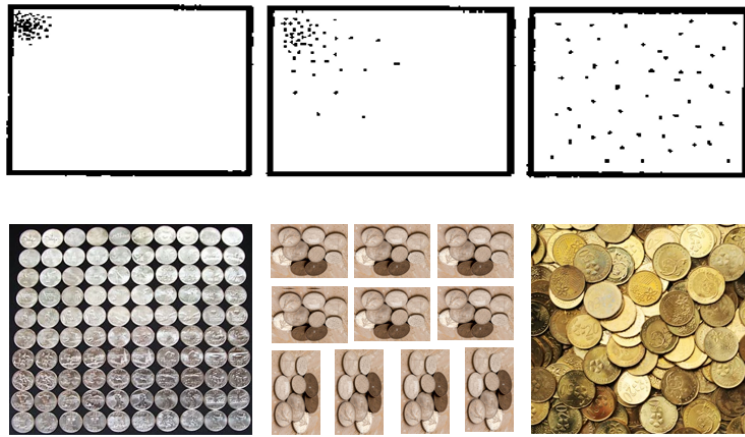


Figure 5: Representation of entropy being limited by structure using gas molecules and coins.

coins are separated and visible, there is no uncertainty as we know the state of every coin. Having no structure at all (image on the right), the number of possible states corresponding to 50 heads and 50 tails is equal to  $10^{29}$ . However, the introduction of some structure, for instance 10 heaps of coins with 5 heads and 5 tails (middle image), will limit the number of possible internal states to 2520.

Though often not realized, the need for structure to limit the number of possible combinations in pattern recognition, the underlying concept of ANNs and LLMs, is overwhelming. Today, large-scale LLMs are being constructed with the amount of parameters in terms Giga ( $10^9$ ), Tera ( $10^{12}$ ), or even Peta ( $10^{15}$ ). While these amounts of parameters may seem large at first sight, e.g., the number of meters in a light-year is  $10^{16}$ , and the total number of atoms in the universe is estimated to be approximately  $10^{80}$ , they are tiny when you start looking at the amount of possible input combinations. Consider for instance an image with 1 megapixel (1000 rows and 1000 columns), and 100.000 different colors. In this case, the total number of possible combinations for a single image would be  $10^{5.000.000}$ . This clearly shows the sheer infinite number of possible combinations, even compared to the total number of atoms in the universe. It is therefore imperative to create structure if one could ever hope to process these near infinite numbers of combinations. Otherwise, even consuming all the energy available in our galaxy would not even be enough to train ANNs to evaluate all possible combinations.

## 4 Modular and Hierarchical Systems

It seems logical that some modular and/or hierarchical structure is needed to limit and control the near infinite possible input combinations in complex systems. Modularity is a cornerstone of engineering, and is generally associated with complexity reduction, and improved robustness for adaptation. Moreover, Hartwell et al. argue that modular structure is a common characteristic of both natural lifeforms and man-made products [8].

### 4.1 The Architecture of Complexity

The concept that *"the architecture of complexity is hierarchy"* originates from Herbert Simon's foundational essay[5], and was further elaborated in his seminal book *"The Sciences of the Artificial"* [6]. Herbert Simon argues that hierarchic systems will evolve far more quickly than non-hierarchic systems of comparable size [5]. He also explored the dynamic properties of hierarchically organized systems and showed how they can be decomposed into subsystems in order to analyze their behavior. In his famous *"Tempus and Hora"* fable, he showed that the evolution of complex forms depends critically on potential

stable intermediate forms.

While Herbert Simon already argued that complex systems in nature are also composed of interrelated subsystems, a very detailed and powerful elaboration of the hierarchical architectures and strategies in biological visual systems was presented by David Van Essen and Charlie Anderson [7]. Van Essen and Anderson explain how the retino-geniculate pathway applies hierarchical sampling strategies and transmission mechanisms [7]. They further explain in great detail the hierarchical architecture and organization in both the neural representations and information flow in the visual cortex.

Recently, the impression may have originated that, within the realm of AI, large-scale LLM's are emerging as monolithic, and therefore non-hierarchical complex systems. However, even within the area of AI, the most complex systems currently available, such as humanoid robots and full self-driving, are still hierarchical modular systems composed of interrelated subsystems. To illustrate this point, we quote the answers regarding this issue of the AI Grok, closely related to one of the leading companies in humanoid robots and full self-driving.

Do monolithic LLM's power Humanoid Robots ?

*No, the logic and control of humanoid robots is not mainly or purely based on Large Language Models (LLMs). LLMs play a growing but supporting role, primarily in high-level reasoning, task planning, and natural language interaction. The core control—especially low-level motion, balance, dexterity, and real-time execution—relies on a mix of traditional robotics techniques, reinforcement learning (RL), model predictive control (MPC), trajectory optimization, and increasingly specialized end-to-end neural networks (often called Vision-Language-Action or VLA models).*

—Grok, April 2026.

Do monolithic LLM's power Full Self-Driving ?

*No, the logic and control of full self-driving (FSD) systems in cars is not mainly or purely based on Large Language Models (LLMs). Like humanoid robots, FSD relies on a hybrid, hierarchical architecture where specialized neural networks handle the core real-time perception, planning, and control. LLMs (or more precisely, their multimodal extensions like Vision-Language Models/VLMs or Vision-Language-Action/VLA models) play an emerging but supporting role—primarily for high-level reasoning, interpretability, scene understanding, and natural-language interaction.*

—Grok, April 2026.

## 4.2 Modular Structures in Software

In hardly any engineering discipline is modular design and architecture so prevalent and omnipresent as in software engineering. It is therefore quite peculiar that AI code generation is currently based on massive non-hierarchical LLM's. Once again, we illustrate this with an answer of the AI Grok.

Do monolithic LLM's power AI Code Generation ?

*Yes, modern AI-based code generation is mainly (and in practice, almost purely) based on Large Language Models (LLMs). The dominant tools and systems people refer to when talking about "AI code generation" — such as GitHub Copilot, ChatGPT, Claude, Cursor, Code Llama, Gemini Code Assist, and similar assistants — all rely on LLMs (or code-specialized variants of them) as their core engine.*

—Grok, April 2026.

Nevertheless, computer programs can be quite complex systems, implying that systems that create these complex computer programs should in general be even significantly more complex. Moreover, we could use the same argument as in Section 3.2 to show that the number of possible combinations to write computer programs with tens of thousands or even millions lines of code becomes quickly near infinite.

The purpose of *Normalized Systems Theory (NST)* [10] has always been to create a deterministic structure to limit uncertainty, and to provide traceability and evolvability. This relationship with the reduction of

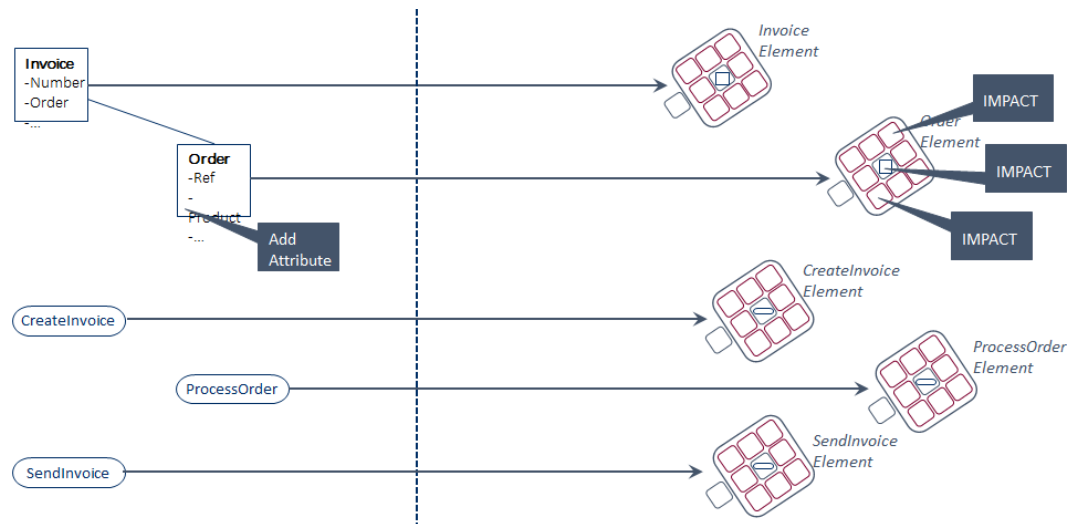


Figure 6: Representation of Normalized Systems elements encapsulating changes.

complexity through structure is clearly reflected in the four NST principles.

- *Separation of Concerns* ensures that structural separation avoids the combined complexity or realization possibilities of both concerns
- *Data Version Transparency* avoids the combined complexity or realization possibilities of different data versions with logic operating on it
- *Action Version Transparency* enables the improvement of logic without the combined complexity or realization possibilities with other logic
- *Separation of States* ensures that structural separation avoids the combined complexity or realization possibilities of processing steps

The elements in NST software architectures [10], as presented in Figure 6, are mainly touted for their ability to encapsulate changes, thereby depleting ripple effects and enabling systems theoretic stability. However, these elements also establish an hierarchical architecture consisting of interrelated subsystems, where the element structures serve to create structural boundaries and to encapsulate possible variations within those boundaries.

## 5 Life, Evolution, and Modularity

Modular and hierarchical structures are clearly present in both artificial and biological systems. However, such structures even seem to be essential and nearly omnipresent in biological systems, leading to the interesting question whether they could be related to the process of evolution itself.

### 5.1 Modular Building Blocks in Life

As explained in Section 4, it has long been recognized that architectures that exhibit functional separation into modules are more robust and amenable to design and adaptation. To illustrate the argument that modular structure is a common characteristic of natural lifeforms and man-made products, we have mentioned the detailed analysis of hierarchical structures and strategies in biological visual systems by Van Essen and Anderson [7].

The presence of modularity or modular structures in natural life forms seems to extend far beyond the

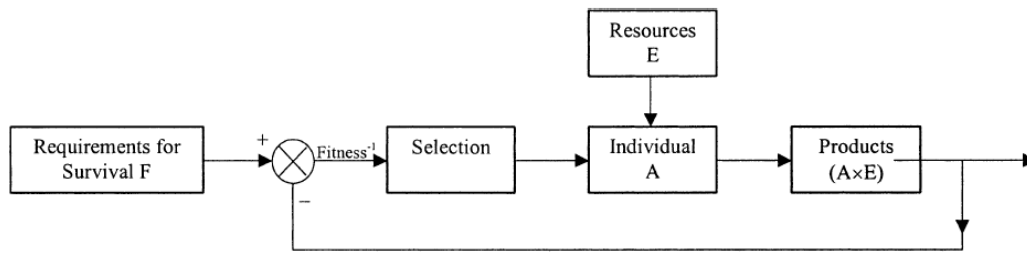


Figure 7: Model representing an individual A transforming a set of resources E in a set of products [9].

hierarchical architectures such as described by Van Essen, all the way to the fundamental building blocks of life, the cells [8]. A consensus seems to exist in cell biology literature that biological cells are highly complex, organized structural and functional units of life. They operate as microscopic, self-sustaining machines containing specialized organelles (like mitochondria, endoplasmic reticulum, and Golgi complex) that manage energy production, protein synthesis, and transport.

The highly organized modular structure of natural lifeforms, corresponding with low levels of entropy as explained in Section 3.2, seems to be so fundamental that NASA scientist *James Lovelock* gave the following answer when asked how he would go about the detection of life on another planet: “I’d look for an entropy reduction, since this must be a general characteristic of life.”. This motivated several researchers to look into the relationship between modularity and the essential processes of natural lifeforms: evolution and natural selection.

## 5.2 On Evolution and Normalization

In their landmark paper, Lipson, Pollack and Suh investigate the relationship between modularity and the response to variation in evolutionary systems [9]. Their model represents an *individual*, subject to selection, as a matrix  $A$  which transforms a set of *environmental resources*  $E$  into an arbitrary set of *products*  $A \times E$  to match a set of *requirements*  $F$ . As schematically represented in Figure 7, the *fitness* selection is based on the residual term  $R = F - A \times E$ . They state that the representation of an individual as a matrix  $A$ , and the equation as a whole, can take on a variety of interpretations. One interpretation is that the vector  $E$  represents genes, the vector  $F$  represents traits, and the transformation matrix  $A$  represents pleiotropic relationships. Another interpretation is that the vector  $E$  represents organic building blocks, the vector  $F$  represents some desired functionalities that must be carried out by the organism to survive, and  $A$  represents how building blocks are composed. The latter interpretation is extremely close to the original starting point of NST, representing a *system* as a matrix transforming a set of modules or *constructive building blocks into a set of functional requirements* [10].

When Lipson et al. [9] examine the relationship between modularity and evolutionary systems, they implicitly assume that modularity corresponds to *modularity done right*, i.e., thoroughly decoupled modules that exhibit high cohesion and low coupling. To quantify modularity, they propose to rearrange the matrix in a near block-diagonal matrix, where the number of blocks corresponds to the number of (sub)systems and is a measure of modularity. The more blocks or decoupled (sub)systems, the higher the modularity. For instance, the matrices in Figure 8 represent respectively six subsystems on the left and two on the right. Once again, this is extremely close to the essential premise — and even the name — of NST, aiming to *normalize the system by normalizing its function-construction transformation matrix* [10] that represents the system.

Lipson, Pollack and Suh find a clear relationship between modularity and the response to variation in evolutionary systems. When the environment changes, a population with higher coupling, i.e., lower modularity or inferior modular structure, suffers harder blows, and needs longer recovery times [9]. We can relate this in a rather straightforward way to the relationship between structure and entropy. *Less*

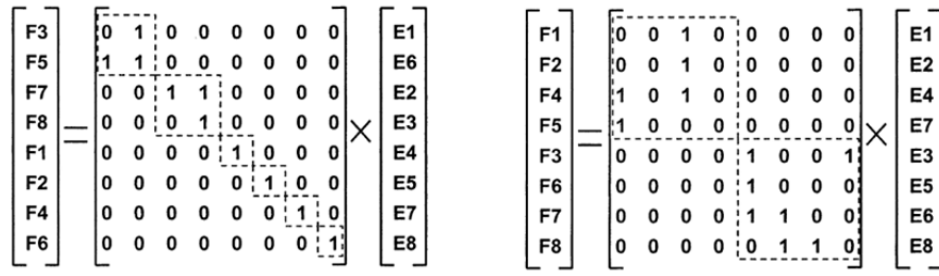


Figure 8: Representation of six decoupled modules (left) and two decoupled modules (right) [9].

*modular structure* implies more entropy, more uncertainty, and *more possible combinations* or possibilities for the variations. Though not all possible combinations are possible for the blocks or subsystems, the total number of combinations for such a block with dimension  $n$  is related, using the *Big O notation*, to the following function.

$$O(2^{(n^2)}) \quad (1)$$

Applying this to the example structures of Figure 8, and using the *Big O* approximation except for the blocks with  $n = 1$  where the entry has to be zero, we get the following numbers:

$$2^4 \times 2^4 = 2^8 \quad \longleftrightarrow \quad 2^{16} \times 2^{16} = 2^{32} \quad (2)$$

Once again, we observe the enormous amounts of possible combinations for systems without, or with limited, modular structure, e.g.,  $2^{1000000}$  for a system with dimension  $n$ .

Lipson, Pollack and Suh also found that modularity arises spontaneously in evolutionary systems in response to variation, and increases with higher variation rates. More specifically, the amount of modular separation is logarithmically proportional to the rate of variation [9]. Modularity does not only enable evolutionary systems to cope better with variation, it even seems to arise as a fundamental long-term consequence of variation and selection.

## 6 Conclusions

The domain of *Artificial Intelligence* goes back to the 1950s. While different technology streams have existed — and still exist — to pursue AI, the main approach today originated from pattern recognition and the mimicking of the processing units in biological systems, i.e., networks neurons. These *Artificial Neural Networks* became a widespread success when they were used to represent and generate language in so-called *Large Language Models*.

Some current issues that these LLMs face, e.g., the occurrence of hallucinations, can be traced back to the early days of pattern recognition in ANNs, where small changes in direct inputs could already lead to completely unexpected results. Such effects are easy to understand, considering that the enormous amount of possible combinations in input data that can lead to strange correlations. The need for structured (pre)processing to avoid these effects was already recognized in the early days of pattern recognition through ANNs.

On a more general level, systems dealing with complex problems need hierarchical and modular structures to control the near infinite number of combinations. Indeed, it can easily be shown that the number of possible input combinations of a single image far exceeds the total number of atoms in the universe. At the same time, the study of entropy shows that the only way to control these enormous amount of possible combinations, and the corresponding uncertainty, is hierarchical structure.

Moreover, it has been shown that such hierarchical systems with decoupled modular subsystems are more robust and amenable to design, and adaptation as they can be tested and improved independently. Unsurprisingly, biological systems also exhibit hierarchical modular structures, both at the level of system architectures, e.g., biological visual systems, and at the level of the basic building block, i.e., biological cells. It has been shown that these modular structures, limiting the number of possible combinations, also enable these biological systems to perform and adapt better under environmental changes.

In the same way, the amount of possible combinations of thousands or even millions of lines of code in software systems far exceeds the total number of atoms in the universe. It is therefore imperative to control this sheer infinite amount of combinations, and the corresponding uncertainty, through modular structures. Unlike real-time AI systems, such as humanoid robots and full self-driving cars, LLM-based code generators still lack these structures today. Therefore, it is crucial to apply LLM-based code generation *within the guardrails of decoupled modular structures* to mitigate the near infinite possibilities that entail uncertainty and the obstinate hallucinations.

## References

---

- [1] McCulloch, Warren S.; Pitts, Walter. "A logical calculus of the ideas immanent in nervous activity". The Bulletin of Mathematical Biophysics. 5 (4): 115–133, 1943.
- [2] Hebb, Donald. "The Organization of Behavior: A Neuropsychological Theory". Wiley, 1949.
- [3] Rumelhart, D. E.; Hinton, G. E.; Williams, R. J. "Learning representations by back-propagating errors". Nature, 323, 533-536, 1986.
- [4] Vaswani, Ashish, Shazeer, Noam, Parmar, Nikki, et al., "Attention is all you need", 31st Int. Conf. on Neural Information Processing Systems, 2017.
- [5] Simon, Herbert, "The Architecture of Complexity", Proceedings of the American Philosophical Society, 1962.
- [6] Simon, Herbert, "The Sciences of the Artificial", MIT Press, 1969.
- [7] Van Essen, David, Anderson, Charlie, "Information processing strategies and pathways in the primate visual system", An introduction to neural and electronic networks, 1995.
- [8] Hartwell, L., et al., "From molecular to modular cell biology", Nature, 1999.
- [9] Lipson, H., Pollack, J. B., Suh, N. P., "On the origin of modular variation", Evolution, 2002.
- [10] Mannaert, H., Verelst, J., De Bruyn, P., "Normalized Systems Theory: From Foundations for Evolvable Software Toward a General Theory for Evolvable Design", NSI Press, 2016.