Athens Journal of Sciences- Volume 12, Issue 1, March 2025 – Pages 59-70

How to Teach Literacy to Artificial Neural Networks Making AI intelligent by Learning from other Disciplines

By Thomas Fehlmann[*] & Eberhard Kranich[±]

What does literacy mean in AI? Generative AI, especially Large Language Models (LLM), use statistical relevance to build responses to prompts. Literacy in education means understanding cause and effect from a text and why one observation follows another. It has to do with the real world and some understanding of how the grounding behaves and works. This kind of learning can be achieved with intelligent systems that combine AI engines with traditional programming, or in terms of the Graph Model of Combinatorial Logic: Observations and Concepts with Lambda Concepts. In conclusion, it is very helpful to listen to other disciplines for making AI intelligent. The respective task list for AI engineers includes, but is not limited to, education and teaching to children and humans.

Keywords: *artificial intelligence, generative pretrained translators, knowledge acquisition transformation, intelligent systems*

Introduction

Kausalai "Kay" Wijekumar, a professor at Texas A&M University, USA, and his team gave a talk at the 26th Annual International Conference on Education, 20-23 May 2024, collocated with Computer Science, on how to teach children to gain literacy by reading. This turned out to be an excellent tutorial on how to make an artificial neural network intelligent.

The problem with the term "Artificial Intelligence" (AI) becomes apparent when you try to translate it into other languages. In German for example, "intelligent" also means "smart", "bright", and "clever"; in Greek it includes "νουσ" which is related to the English "nous", but is probably better explained in the Iliad, where in Book 18 the (female) androids who helped Hephaistos to craft Achilles' new suit of armor are called not only intelligent, but also "νουσ". For Homer, this meant that these androids were capable not only of doing what Hephaistos said, but also of doing what he meant (Homer 1898).

Today's AI is great at intelligence in the sense of data collection, which is what the U.S. federal intelligence agencies like the FBI and CIA are called, but not at understanding what is really happening, what it means.

[*]Senior Researcher, Euro Project Office AG, Switzerland.
[±]Senior Researcher, Euro Project Office AG, Switzerland.

Literature Review

Wijekumar's research focus on how children and pupils can become literate, which means, not only able to read and write, but understanding cause and effect, identify the ideas developed by the protagonists, summarizing the intend of the authors, able to explain what causes their behavior, and predicting consequences. As an example, we cite a recent paper about the importance of inference generation in the meaning-making process (Rice & Wijekumar 2024). However, in educational science and practice, there exists a large literature and professional associations that discuss best teaching practices that we cannot fully reference here.

More directly related to AI are contributions from other disciplines such as Theoretical Computer science that also try to shed light on how the human brain thinks. Engeler published in 2019 that the *Graph Model of Combinatory Logic* is an excellent model for "how does the brain think" (Engeler 2019). The graph model is an algebraic way of describing neural networks, i.e., directed graphs that contain loops. *Artificial Neural Networks* (ANN) do not contain loops, at least not the most prominent variety, the *Feedforward Neural Networks* (FNN). They consist of an input layer, several hidden layers and an output layer. The graph model, as combinatory logic, is Turing-complete (Turing 2004) and thus contain loops that potentially loop forever. In Computer Science, such fixpoint operators play a substantial role. It cannot be determined whether they ever stop looping. For the brain, Engeler introduces the concept of *Controlling Combinators*; loops that converge towards some *Attractor*, a construct in the neural network, aka the brain, that corresponds to a skill. As examples, he constructs controlling combinators for a violinist and for a mathematician (Engeler 2019).

The authors have shown in several papers that this graph model also serves as a general model for knowledge, with the particularity that it not only models cause-effect relations, but also allows modeling abstract concepts such as Turing programs. (Fehlmann & Kranich 2024) For practical work such as requirements engineering for AI-enabled systems, this provides the opportunity to add so-called lambda concepts that enforce rules and guarantee safe and secure operations. (Fehlmann & Kranich 2023).

Regarding classical AI based on *Large Language Models* (LLM) or any other variant of ANN, we refer to the rapidly evolving literature.as an entry point, Gerven & Bothe's classification might be a good start (Gerven & Bohte 2017). Despite the rapid advances occurred in the past few years, AI is still not able to solve basic tests found for instance in human intelligence tests, see Chollet's ARCathon, now ARC Price (Chollet 2019).

On the other hand, a team around Panigrahi at Princeton University has recently published work showing how to identify skill areas in an ANN. They use a technique called "grafting". (Panigrahi et al. 2023). This greatly simplifies the training of an ANN by allowing you to focus on some specific skill area without touching other parameters outside that skill area, and grafting allows you to implant these newly trained skills into other ANNs.

Finally, we must mention the ISO/IEC 19761 COSMIC standard for modeling software (COSMIC Measurement Practices Committee 2020). It turns out that the

way software is modeled in this standard is ideally suited to the needs of intelligent systems because the model focuses on data movements, moving groups of data that can be easily identified with some specific knowledge. This allows tracing what happens to key performance indices such as reliability when data from AI engines is processed (Fehlmann & Kranich 2024).

Methodology and Methods

Intelligent systems are software-driven systems that incorporate one or more AI engines and interact with their environment either through sensors and actuators, as on the Internet of Things (López et al. 2013), or any natural or logical language. This interaction allows them to perceive whether their actions were successful or not, and to learn from failures. As such, they share some similarities with living beings, especially children and adolescents who are learning new skills (Russell 1948).

The aim of this paper is to investigate whether principles that are used in education by teachers are suitable as a blueprint for building intelligent systems, and we claim, they are.

What is Knowledge?

The graph model of combinatory logic is an interesting model for representing knowledge because the directed graphs used represent neural networks. Thus, there is a direct connection between ANN in AI, the biological brain, and this model (Fehlmann & Kranich 2024).

The constituting elements of the graph model are *Combinators*, defined as sets of arrow terms of the form (1):

$$x_j \to y \tag{1}$$

The nodes of origin in the graph x_j are a selection of *Observations*, selected by the choice function j, and the y represent the target node, the node receiving a stimulus from the x_j, also interpreted as observed effect. We call proper arrow terms *Concepts*. We claim that these combinators, consisting of observations and concepts, represent knowledge (Fehlmann & Kranich 2024). This definition is highly recursive, because you can observe concepts. Thus, arrow terms can contain other arrow terms including concepts.

If this powerset is based on the null set, weights in nodes play no role in equation (1). If the powerset is based on some non-empty set of observations referring to some real world, then you can add weights to the nodes for describing their impact on the target node and this powerset describes the behavior of both artificial and natural neural networks (Engeler 2019).

Knowledge is combined as follows. Let M and N be any combinators, then you can apply M to N:

$$M \bullet N = \{b | \exists \mathrm{a_i} \to b \in M;\ a_i \in N\} \tag{2}$$

Equation (2) makes the graph model an algebra (Engeler 1981). The algebra is Turing-complete because it is possible to define *Lambda Terms* that introduce a variable x in M, allowing for an application of M to some argument N (Barendregt & Barendsen 2000):

$$\lambda x.M \bullet N \tag{3}$$

In case of equation (3), N replaces all occurrences of $x\ in\ M$. For formal definitions, consult (Fehlmann 2020, p. 5). In the graph model, Lambda terms contain no observations; thus, it is a concept. It has the form of a complicated structural element whose application does not depend on its nodes' weights (Fehlmann 2016, p. 326ff). For this characteristic, we call it *Lambda Concept*. For a proof of Barendregt's theorem for the graph model, see Fehlmann (Fehlmann, 1981).

Barendregt's Lambda calculus means that programmable terms exist in the context of knowledge, making fixed rules part of general knowledge. For humans, this is nothing surprising; for machines, it is good to know that observation-independent rules exist like those used in social interactions between humans. Like in computer programming, you can write Lambda terms that contain no open variable terms, or you can write programs that process observations. And you can write Lambda terms that loop forever, e.g., fixpoint operators (Fehlmann & Kranich 2022).

Controlling Combinators

The basic characteristic of ANN is that there are no loops allowed from the input layer to the output layer. There are hidden layers in between but even so, they do not provide feedback. This holds not only for *Feedforward Neural Networks* (FNN); *Recurrent Neural Networks* (RNN) are bi-directional ANNs that allow the output from some nodes to affect subsequent input to the same nodes (Gerven & Bohte 2017).

In contrary, the natural brain, as well as Engeler's graph model of combinatory logic, have feedback loops by dendrites that can stimulate or attenuate responses. As in every traditional computer program, such loops can loop forever and damage neuronal reactions. Thus, the brain has mechanisms that Engeler modelled as controlling combinators. These are crucial for further developing artificial intelligence and better approximate human intellectual capabilities (Engeler 2019).

The concept of *Control* involves a *Controlling Operator* $\mathbf{C}$ which acts on a controlled object X by application $\mathbf{C} \bullet X$. Control means that the knowledge represented by X is completely known and described. It is a similar approach to establishing a fixpoint.

Accomplishing control can be formulated by:

$$\mathbf{C} \bullet X = X \tag{4}$$

The equation (4) is a theoretical statement, usually resulting in an infinite loop process. For solving practical problems, X must be approximated by finite subterms.

The *Control Problem* is solved by a *Control Sequence* $X_0 \subseteq X_1 \subseteq X_2 \subseteq \cdots$, a series of finite subterms, determined by (5):

$$X_{i+1} = \mathbf{C} \bullet X_i, i \in \mathbb{N} \tag{5}$$

starting with an initial X_0. This is called *Focusing*. The details can be found in Engeler (Engeler 2019, p. 299). The controlling operator $\mathbf{C}$ gathers all faculties that may help in the solution. Like a fixpoint combinator in combinatory logic, controlling operators are a structural element, not a single pair of observations and observed effect. The control problem is a repeated process of substitutions.

Within this setting, it is possible to define models for reasoning (Engeler 2019), problem solving, and software and systems testing (Fehlmann & Kranich 2019). Moreover, since the model includes both algorithms and cause-effect statements, it can be used for specifying requirements for intelligent systems that sometimes must adapt to the environment and sometimes follow strict rules, for instance when legal compliance, or safety, is a strict requirement (Fehlmann & Kranich 2023). While generative AI can generate any idea, reasonable or not, these algorithmic requirements, called lambda concepts, ensure predictable behavior from an intelligent system.

Intelligent systems thus are always a mix of controlling combinators and generative AI. Thus, they behave partly like an ordinary program, executing rules, and partly they interpret their environment by the data received.

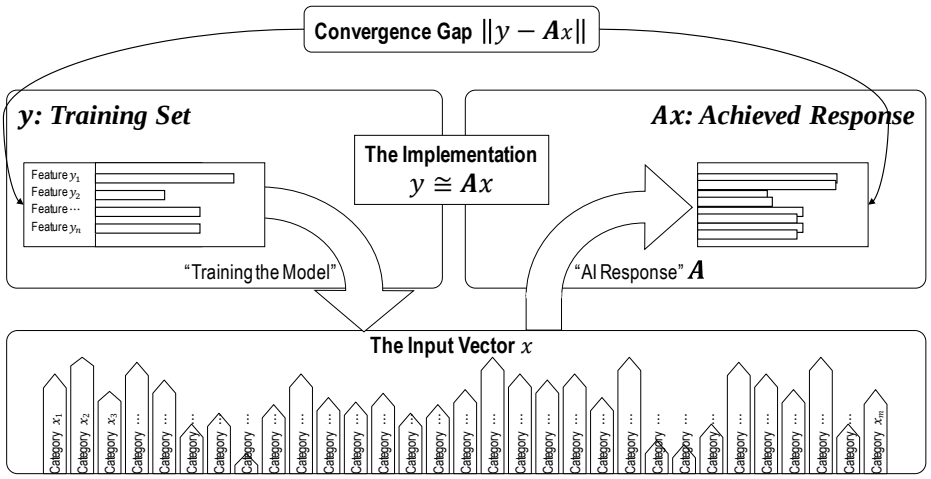
The Convergence Gap

Controlling combinators need a metric that tells them whether the equation (5) actually results in a controlling sequence. Not only in AI, but this is also the most difficult question to solve: whether a change imposed on a system improves performance or not.

The principle of *Deep Learning* (Figure 1) is similar between humans and machine – it means, adjusting a transfer function $\boldsymbol{A}$ which, given an input vector $\boldsymbol{x}$, produces a response $\boldsymbol{Ax}$ that is sufficiently close to the correct answer; correct in the sense that it matches the features of the training set.

For an ANN, which here corresponds to the neural network $\boldsymbol{A}$, the weights of the nodes is adjusted so that the response vector $\boldsymbol{Ax}$ matches the training features $\boldsymbol{y}$ as best as possible. The difference between $\boldsymbol{y}$ and $\boldsymbol{Ax}$ is called *Convergence Gap*. This difference is calculated by the Gauss' method of the least squares (Stigler 1981).

Figure 1. *The Principle of Deep Learning*

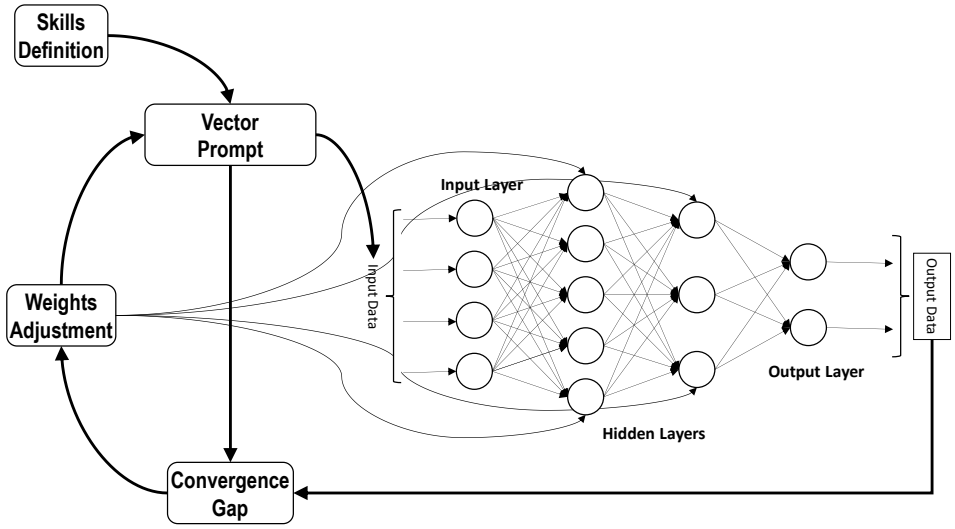


Implementing Controlling Combinators

Although neural networks are well capable of learning algorithmic programming (Fehlmann & Kranich 2024), implementing controlling combinators in an ANN is difficult by the lack of feedback loops. Thus, the feedback loops have to be provided by some embedding environment, for instance Phyton that runs most of the ANNs.

The requirements for a controlling combinator include the ability to deal with potentially infinite calculations and repetitions executed by an ANN. This includes the ability to stop the ANN's learning cycle once the precision achieved is good enough. This is measured by the convergence gap but requires some upfront definition of what means "good enough". Usually, in all kinds of undertaking, this is the most difficult step.

Figure 2. *Controlling Combinator Implementation Scheme*



According to

Figure 2, a controlling combinator implements at least the following three functional processes: prompting, convergence gap calculation, and parameter adjustments in the ANN, here represented by its input and output layer, plus two or more hidden layers (Gerven & Bohte 2017).

The basic principle of a controlling combinator has some similarities with deep learning. The controlling sequence is implemented by adjustments to the parameters of the AI engine, e.g., the weights in an ANN. However, contrary to deep learning, the controlling combinator can adapt the criteria for good learning to what the systems learned before. The controlling combinator consists of three functional processes:

- The first functional process supplies the training set to the respective part of the AI engine. This is called *Skills Definition*. This functional process might need to access one or several knowledge bases, and the Internet.
- The second functional process measures the success rate of the controlling sequence and is called *Convergence Gap*. This functional process needs access not only to knowledge bases but also to operational data representing an actual status of a system, such that it can compare responses from the AI to factual evidence.
- The third functional process adjusts the parameters of the AI engine (e.g., the weights of an ANN) according to some suitable learning strategy. The controlling combinator repeats operations until the convergence gap closes sufficiently well.

The convergence gap usually does not reach zero because that would mean we have a deterministic system that might violate the undecidability of predicate logic, consult Gödel (1931) and Raatikainen (2020).

Applying Controlling Combinators to Literacy

Wijekumar's *Knowledge Acquisition Transformation* (KAT) framework consists of the following six steps:

1) Learn Vocabulary – basic reading skills.
2) Text Reading – text in full detail.
3) Guide Thinking – try to identify ideas.
4) Important Ideas – which ideas matter?
5) Summarize Main Ideas – reproduce relevant ideas.
6) Extrapolate Inferences:

    - Get the mechanics of the plot;
    - Become able to reproduce it.

For step 3 – Guide Thinking – Wijekumar proposes a logical schema, the Cause-Problem-Solution principle. This can be understood as a logical framework that links elements of the plot together – such as in book 9 of the Iliad, where Achilles' rage does not allow him returning under the command of Agamemnon. The

solution of the problem is sending Patroclus instead. This then causes new problems (Homer 1898).

To implement the KAT framework, we need a controlling combinator for each of the six steps:

1) Pre-Trained LLM – You can start with any pre-trained LLM.
2) Fine-tune LLM with text – You should train the LLM with the specific text, e.g., Homer, or some Netflix plot.
3) Extract **C**ause-**P**roblem-**S**olution triplets – Do a structural analysis of what you read. This involves that the intelligent system uses a logical model – the C-P-S triplets – to collect cause-effect ideas from the text.
4) Logically combine C-P-S triplets – we might need to use the services of a logical engine to do that.
5) Find proofs – sequences of logically dependent C-P-S triplets that lead to some conclusion that fits the plot.
6) Let the LLM explain the plot as a logical sequence.

Basically, we adapt these six steps to understand the learning cycles of an intelligent system. While step one and step two (pre-learning and fine-tuning) are following today's standards in deep learning, step three involves a logical structure that exists outside the LLM. It will be used for training the LLM in logical skills. In some sense, the traditional logic represented by the C-P-S triplets complements the language skills available in the LLM. This makes, in turn, the responses of the LLM explainable. The fourth and the fifth step are well controllable by the Controlling Combinator in the sense, that a Convergence is easily calculated that identifies correct logic and whether the resulting effect have some relevance to the original text.

Intelligent means in this context that some continuous learning occurs after initial pre-training and fine-tuning of the large language models (LLM). It's an interesting question that we cannot answer yet, whether original texts like the Iliad or modern, derived plots, e.g., from Netflix, work better. The full KAT implementation design can be found in a collection of Intelligent System designs by Euro Project Office (Fehlmann 2024).

In any case, intelligent systems are a mix of traditional, algorithmic programming and generative AI engines based on ANN like the currently emerging LMM. No design for an AI engine currently supports self-paced learning without help from algorithmic programming, usually in Python. For this, we introduce the software modelling approach following ISO/IEC 19761.

The ISO/IEC 19761 Model of Software (COSMIC)

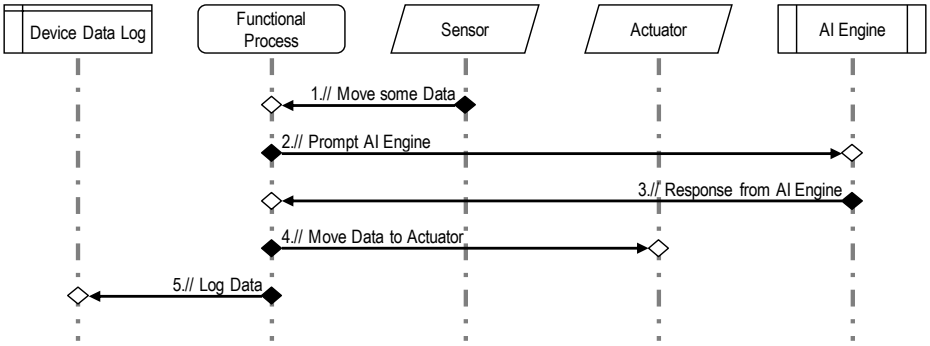
As an ideal tool for designing intelligent systems, we use the ISO/IEC 19761 model of software (ISO/IEC 19761, 2019). This model introduces the concept of *Data Group* (COSMIC Measurement Practices Committee 2020, p. 13), a group of logically connected data describing objects of interests from the user's perspective. The data groups are moved between a functional process and devices, other application, or permanent storage. The model distinguishes four types of data movements:

Entry, eXit, Read and Write. It is easy to identify the data groups as specific sets of knowledge that are moved from one object of interest to another. Thus, data movements carry the knowledge acquired by the intelligent system.

The AI part of the intelligent system is modelled as "another application". Its response might be subject to some uncertainty.

Similar to UML diagrams, we represent software as *Data Movement Maps*, identifying objects of interest with lifelines and let the data movements execute between them. This yields a rough description of the software modelled. It can be used to measure reliability of an intelligent system (Fehlmann & Kranich 2024) but does not specify the exact execution of loops in a program. Nevertheless, it carries enough detail to allow for the representation of controlling combinators that support intelligent systems when learning behavior and details of its environment.

Figure 3. *Sample Data Movement Map Driven by an AI Engine Application*



Results

This research started by end of May 2024 and will be continued for some time. Preliminary results suggest learning from other disciplines is probably the king's way towards intelligent systems. It is not helpful to dismiss older attempts as "failures" only because they did not meet expectations – that always were too high, usually – and because current LLMs outperform all previous attempts to handle natural language.

A Sample Implementation of KAT

Currently, only a design exists for the KAT implementation in the form of a commented data movement map. We are looking for partners to implement them in a real system. KAT is not a small intelligent system. With its six times three functional processes, implementing six controlling combinators for learning reading skills in the sense of literacy, it counts a few hundred data movements, i.e., COSMIC function points. Its reliability – a performance measurement (Fehlmann & Kranich 2024) – depends on the choices for the thresholds for the six convergence gaps (Hunt 2003).

Applying the Method to Other Challenges

Conceptualization is a challenge for all kinds of LLM. The ARC Price challenge (formerly ARCathon) is still open (Chollet 2019). So far, no approach is known that lets an LLM learn the rather basic conceptualization skills that are needed to solve the ARC Price puzzles, while an average human is easily capable to perceive the rules behind the puzzling drawings and thus solving the ARC Price challenge.

Our method to train an LLM with logical patterns such as the S-P-S triplets should also work for the ARC Price challenge.

Discussion

The big challenge in our approach is getting the convergence gap right. This is what made Wijekumar's work so attractive for us; we learned from her work how to set up suitable convergence gaps for our learning task.

It is not obvious from the design that LLMs can be efficiently trained to do logic. It may well be that another AI engine is better suited for logical reasoning. On the other hand, an ANN is inherently capable of logic because of the shared representation of knowledge (Fehlmann & Kranich 2024) and Barendregt's Lambda theorem. Nevertheless, it might be a different approach than just implementing arrow terms for logical deduction in a neural network. Humans, for example, often do logic by combining language skills and visual imagination of logic rules and logical deduction (at least the author does), but the approaches are likely to be different for each individual.

Conclusions

Making AI intelligent is still a big challenge. We believe that while the impact of LLMs and large ANNs is great and their achievements are stunning, there are many open questions. One is called "Explainable AI" and refers to the problem that it is very difficult to understand an AI decision. Somehow it relates to the training set, its possible biases and shortcomings. Intelligent systems, as we propose them, should be able to combine AI reasoning with adherence to rules, regulations, and safety-first principles, while at the same time improving the system by using control combinators for continuous learning.

Acknowledgments

The authors would like to thank Dr. Kay Wijekumar and her team for their talk at the 26th Annual International Conference on Education, 20-23 May 2024, collocated with Computer Science, in Athens, on how to teach children to gain literacy by reading.

References

Barendregt H, Barendsen E (2000) *Introduction to Lambda Calculus.* Nijmegen: University Nijmegen.

Chollet F (2019) *On the Measure of Intelligence.* Cornell University, Ithaca, NY: arXiv: 1911.01547 [cs.AI].

COSMIC Measurement Practices Committee (2020) *COSMIC Measurement Manual for ISO 19761 – Version 5.0 – Part 1-3.* Montréal: COSMIC Measurement Practices Committee.

Engeler E (1981) Algebras and Combinators. *Algebra Universalis* 13: 389–392.

Engeler E (2019) Neural algebra on "how does the brain think?". *Theoretical Computer Science* 777: 296-307.

Fehlmann TM (1981) *Theorie und Anwendung der Kombinatorischen Logik.* (Theory and Application of Combinatorial Logic). Zürich, CH: ETH Dissertation 3140-01.

Fehlmann TM (2016) *Managing Complexity – Uncover the Mysteries with Six Sigma Transfer Functions.* Berlin, Germany: Logos Press.

Fehlmann TM (2020) *Autonomous Real-time Testing – Testing Artificial Intelligence and Other Complex Systems.* Berlin, Germany: Logos Press.

Fehlmann TM (2024) *Intelligent Systems.* [Online] Available at: https://web.tresorit.com/l/AXX78#FaBkGqfY2cF_JsVmX70_ng.

Fehlmann TM, Kranich E (2019) Testing Artificial Intelligence by Customers' Needs. *Athens Journal of Sciences* 6(4): 265–286.

Fehlmann TM, Kranich E (2022) The Fixpoint Combinator in Combinatory Logic - A Step towards Autonomous Real-time Testing of Software? *Athens Journal of Sciences* 9(1): 47–64.

Fehlmann TM, Kranich E (2023) *Requirements Engineering for Cyber-Physical Products.* Systems, Software and Services Process Improvement. EuroSPI 2023 ed. Grenoble: Communications in Computer and Information Science, Springer, Cham.

Fehlmann TM, Kranich E (2024) Measuring Knowledge - An Attempt to Define a Measurement Principle for Learning Machines. *Athens Journal of Sciences* (forthcoming).

Fehlmann TM, Kranich E (2024) *The Neural Algebra and its Impact on Design and Test of Intelligent Systems.* Palermo, IHSI 2024 Open Access, AHFE International, USA.

Fehlmann TM, Kranich E (2024) A General Model for Representing Knowledge - Intelligent Systems Using Concepts. *Athens Journal of Sciences* 11(3): 181–198.

Gerven Mv, Bohte S (2017) *Artificial Neural Networks as Models of Neural Information Processing.* Lausanne: Frontiers Media.

Gödel K (1931) Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. (On formally undecidable theorems of Principia Mathematica and related systems I). *Monatshefte für Mathematik und Physik* 38(1): 173–198.

Homer (1898) *The Iliad.* London, New York, and Bombay: Longmans, Green and Co.

Hunt D (2003) The concept of knowledge and how to measure it. *Journal of Intellectual Capital* 4(1): 100–113.

ISO/IEC 19761 (2019) *Software engineering - COSMIC: a functional size measurement method.* Geneva, Switzerland: ISO/IEC JTC 1/SC 7.

López TS, Ranasinghe DC, Harrison M, McFarlane D (2013) Using Smart Objects to build the Internet of Things. *IEEE Internet Computing* (to appear).

Panigrahi A, Saunshi N, Zhao H, Arora S (2023) *Task-Specific Skill Localization in Fine-tuned Language Models.* Cornell University, Ithaca, NY: arXiv:2302.06600v2 [cs.CL].

Raatikainen P (2020) Gödel's Incompleteness Theorems. In EN Zalta (ed.), *The Stanford Encyclopedia of Philosophy.* s.l.:s.n.

Rice M, Wijekumar K (2024) Inference Skills for Reading: A Meta-Analysis of Instructional Practices. *Journal of Educational Psychology* 116(4).

Russell B (1948) *Human Knowledge: Its Scope and Limits.* Sixth impression 1976 ed. New York, NY: George Allen & Unwin (Publishers) Ltd, London, UK.

Stigler SM (1981) Gauss and the Invention of Least Squares. *The Annals of Statistics* 9(3): 465–474.

Turing A (2004) Computing Machinery and Intelligence. In *The Turing Test: Verbal Behavior as the Hallmark of Intelligence*, 67. Cambridge, MA: The MIT Press.