The Graph Model of Combinatory Logic as a Model for Explainability

Thomas Fehlmann
Euro Project Office AG
8032 Zurich, Switzerland
E-mail: thomas.fehlmann@e-p-o.com

Eberhard Kranich
Euro Project Office
47051 Duisburg, Germany
E-mail: eberhard.kranich@t-online.de

***Abstract*—Directed graphs such as neural networks can be described by arrow terms linking a finite set of incoming nodes to some response nodes. Scott and Engeler [1] have shown that its powerset is a model for combinatory logic. This algebra is called Graph Model of Combinatory Logic. Combinatory logic is Turing-complete; thus, the model explains both traditional programming as well as neural networks such as the brain. The graph model would yield a performant AI-tool if used as a blueprint for implementing AI. A chain of thoughts would come for free, and explainability with it. However, its performance would make such a tool impractical and useless. We propose a combined approach for adding explainability to AI. It is the strategy humans use when they try to explain their ideas. First, we use the generative power of neural networks to produce an idea or solution. Next, we create a chain of thoughts that explains such ideas to others. AI could follow the same strategy. Anything generated by an AI engine can be analyzed as a sequence of set of arrow terms that explain the line of thinking, provided the AI engine had been trained properly. Improper training, biases, and hallucinations would become detectable. Since the target is known, guided search can find suitable arrow terms in predictable time. The architecture of this proposed AGI engine consists of three distinct elements: a well-trained artificial neural network, a deduction engine for the arrow term sets, and a search engine for fact checking.**

Keywords—Chain-of-Thought (CoT); Artificial General Intelligence (AGI); Artificial Neural Networks (ANN); Combinatory Logic; Quality Function Deployment (QFD).

I. INTRODUCTION

A. Short History of AI and a its Philosophical Background

In the early 20th century, there were some shocking events taking place in mathematical logic and natural science. Gödel [2], when trying to solve some of Hilbert's 23 problems, detected that predicate logic, something with a long history dating back to the ancient Greeks, is undecidable. This insight gave birth to theoretical computer science, including the theory of computation, founded by Turing [3]. For a modern compilation, see Raatikainen [4].

Schönfinkel and Curry [5] developed *Combinatory Logic* to avoid the problems introduced when using logical quantifiers, and Church invented *Lambda Calculus* as a rival formalism [6]. Scott and Engeler developed the *Graph Model* [1], based on *Arrow Terms*, and proved that this is a model of combinatory logic. This means that you can combine sets of arrow terms to get new arrow terms, and that combinators, accelerators, and constructors can be used to create new elements of algebra.

Graphs in the form of neural networks appeared already at the origins of *Artificial Intelligence* (AI). Its first instantiation in modern times was the *Perceptron*, a network of neurons postulated by Rosenblatt [7]. It later became a directed graph [8]. Rosenblatt was also the first who postulated concepts, among perception and recognition, as constituent parts of AI [7, p. 1].

Since its origins, AI has experienced difficulties; however, today it seems to have become mainstream as far as there are many AI applications that provide value for the user. In some areas, training an AI model is much simpler and more rewarding than finding and programming an algorithm.

AI-powered visual recognition systems excel in recognizing and classifying objects, following the ideas established by Rosenblatt [7]. However, they are weak at recognizing temporal dependencies and unable to combine learnings, despite attempts to develop methods with sequential data and the ability to capture temporal patterns. AI lacks what humans use in such cases: a concept.

Logical skills such as inference and deduction provide quite a challenge, as exemplified by the ARC Price challenge, a sort of intelligence test for AI models, proposed by Chollet [9]. A *Large Language Model* (LLM) easily summarizes texts or books but it still does not understand what is written in it, in the sense that the US National Council of English Teachers calls Literacy, see [10], [11].

Regarding LLM or any other variant of *Artificial Neural Networks* (ANN), we refer to the rapidly evolving literature. As an entry point, Gerven & Bothe's classification might be a good start [12]. *Natural Neural Networks*, in analogy to ANNs, are abbreviated by NNN.

IBM defines *Explainable Artificial Intelligence* (XAI) as a set of processes and methods that allows human users to comprehend and trust the results and output created by machine learning algorithms [13]. This is a bold attempt to use statistical correlations as a basis for reasoning. From a theoretical perspective, this is unlikely to work, because of Gödel [2]; however, from an engineering prospective, it is an attempt to work around undecidability. Dallanoce compiled a list of available processes and methods for XAI [14].

Artificial General Intelligence (AGI) is a type of artificial intelligence (AI) that falls within the lower and upper limits of human cognitive capabilities across a wide range of cognitive tasks. The creation of AGI is a primary goal of AI research and companies such as OpenAI and Meta, but what exactly AGI refers to is controversial [15].

B. Research Questions

The aim of this paper is to recall prior work in logic and AI to understand how neural networks work. To do this, we investigate into the following three research questions:

RQ 1: How are neural networks and especially ANNs linked to the graph model?
RQ 2: Does a chain of thoughts relate to a sequence of arrow schemes?
RQ 3: Can arrow schemes explain AI?

The motivation for this is that we are currently experiencing the fourth AI hype in sixty years and that its acceptance in society is currently transitioning from admiration to rejection. Because the nature of AI is poorly understood not only by society but also by the AI research community. We believe that the graph model is an excellent way to understand what intelligence is, both natural and artificial. However, it is not an answer to how to construct XAI.

C. Paper Structure

We first explain combinatory logic (section II) and the motivation for building a model (section III). Then we compare ANNs with graphs and explain how arrow schemes represent what an ANN does and have an outlook on the architecture of intelligent systems (section IV).

II. COMBINATORY LOGIC

Here has been a lack of attention and consequently of publications on *Combinatory Logic.* Nevertheless, it explains quite a bit what artificial intelligence can do and what not.

A. Combinatory Logic and Axiom of Choice

Combinatorial Logic is a notation that eliminates the need for quantified variables in mathematical logic, and thus the need to explain what the meaning of existential quantifiers $\exists x \in M$ is, see Curry [5] and [16]. Eliminating quantifiers is an elegant way to avoid the *Axiom of Choice* [17] in its traditional form. Combinatory Logic can be used as a theoretical model for computation and as design for functional languages (Engeler [18]); however, the original motivation for combinatory logic was to better understand the role of quantifiers in mathematical logic.

It is based on *Combinators* which were introduced by Schönfinkel in 1920. A combinator is a higher-order function that uses only functional application, and earlier defined combinators, to define a result from its arguments.

The combination operation is denoted as $M \bullet N$ for all combinatory terms M, N. To make sure there are at least two combinatory terms, we postulate the existence of two special combinators **S** and **K**.

They are characterized by the following two properties (1) and (2):

$$\mathbf{K} \bullet P \bullet Q = P \tag{1}$$

$$\mathbf{S} \bullet P \bullet Q \bullet R = P \bullet Q \bullet (P \bullet R) \tag{2}$$

P, Q, R are terms in combinatory logic. The combinator **K** acts as projection, and **S** is a substitution operator for combinatory terms. Equations (1) and (2) act like axioms in traditional mathematical logic.

Like an assembly language for computers, or a Turing machine, the **S**-**K** terms become quite lengthy and are barely readable by humans, but they work fine as a foundation for computer science. The power of these two operators is best understood when we use them to define other, handier, and more understandable combinators.

The identity combinator for instance is defined as

$$\mathbf{I} := \mathbf{S} \bullet \mathbf{K} \bullet \mathbf{K} \tag{3}$$

Indeed, $\mathbf{I} \bullet M = \mathbf{S} \bullet \mathbf{K} \bullet \mathbf{K} \bullet M = \mathbf{K} \bullet M \bullet (\mathbf{K} \bullet M) = M$. Association is to the left. Moreover, **S** and **K** are sufficient to build a Turing-machine. Thus, combinatory logic is Turing-complete. For a modern proof, consult Barendregt [19, pp. 17-22].

B. Functionality by the Lambda Combinator

Curry's *Lambda Calculus* [20] is a formal language that can be understood as a prototype programming language. The **S**-**K** terms implement the lambda calculus by recursively defining the *Lambda Combinator* $\mathbf{L_x}$ for a variable x as follows:

$$\begin{gathered} \mathbf{L_x} \bullet x = \mathbf{I} \\ \mathbf{L_x} \bullet Y = \mathbf{K} \bullet Y \text{ if } Y \text{ different from } \mathbf{x} \\ \mathbf{L_x} \bullet M \bullet N = \mathbf{S} \bullet \mathbf{L_x} \bullet M \bullet \mathbf{L_x} \bullet N \end{gathered} \tag{4}$$

The definition holds for any term **x** of combinatory logic. Usually, on writes suggestively $\lambda x.M$ instead of $\mathbf{L_x} \bullet M$, for any combinatory term M. *Lambda Terms* $\lambda x.M$ offer the possibility of programmatic parametrization. Note that $\lambda x.M$ is a combinatory term, as proofed by (4), and that this introduces a kind of variable in combinatory logic with a precisely defined binding behavior.

The Lambda combinator allows writing programs in combinatory logic using a higher-level language. When a Lambda term gets compiled, the resulting combinatory term is like machine code for traditional programming languages.

C. The Fixpoint Combinator

Given any combinatory term Z, the *Fixpoint Combinator* **Y** generates a combinatory term $\mathbf{Y} \bullet Z$, called *Fixpoint of* Z**,** that fulfills $\mathbf{Y} \bullet Z = Z \bullet (\mathbf{Y} \bullet Z)$. This means that Z can be applied to its fixpoint as many times as wanted and still yields back the same combinatory term.

In linear algebra, such fixpoint combinators yield an eigenvector solution $\mathbf{Y} \bullet Z$ to some problem Z.

According to Barendregt in his textbook about Lambda calculus [19, p. 12], the fixpoint combinator can be written as

$$\mathbf{Y} := \lambda f.\left(\lambda x.f \bullet (x \bullet x)\right) \bullet \left(\lambda x.f \bullet (x \bullet x)\right) \quad (5)$$

Translating (5) into an **S**–**K** term demonstrates how combinatory logic works, see the authors' paper from 2022 [21].

When translated into arrow terms, the fixpoint combinator contains loops. Fixpoint operations are related to infinite loops, programming constructs that never end in some normal form. Applying **Y**, or any equivalent fixpoint combinator to a combinatory term *Z*, usually does not terminate. An infinite loop can occur, and must sometimes occur, otherwise Turing would be wrong and all finite state machines would reach a finishing state [3].

III. The Graph Model of Combinatory Logic

The graph model is a versatile model for knowledge in all its instantiations. It is highly recursive and Turing-complete, which means, it also describes conventional algorithmic programming.

A. A Logic Needs a Model

A *Model* for a logical structure is a set-theoretic construction that has the properties postulated for the logic and can be proved to be non-empty. Then it means that logic makes sense as far as it describes some structure that really exists. If a non-empty model exists, then the logic exists in the sense that it can be used to prove something about the model.

Let $\mathcal{L}$ be a non-empty set. Engeler [1] defined a *Graph* as the set of ordered pairs:

$$\langle \{a_1, a_2, \dots, a_m\}, b \rangle \quad (6)$$

with $a_1, a_2, \dots, a_m, b \in \mathcal{L}$. We write $\{a_1, \dots, a_m\} \to b$ for the ordered pair to make notation mnemonic, i.e., referring to directed graphs, and call them *Arrow Terms*. These terms describe the constituent elements of directed graphs with multiple origins and a single node. We extend the definition of arrow terms to include all formal set-theoretic objects recursively defined as follows:

$$\begin{gathered}\text{Every element of } \mathcal{L} \text{ is an arrow term.} \\ \text{Let } a_1, \dots, a_m, b \text{ be arrow terms.} \\ \text{Then } \{a_1, \dots, a_m\} \to b \text{ is also an arrow term.}\end{gathered} \quad (7)$$

The left-hand side of an arrow term is a finite set of arrow terms, and the right-hand side is a single arrow term. This definition is recursive. Elements of $\mathcal{L}$ are also arrow terms. The arrow, where present, should suggest the ordering in a graph, not logical imply.

B. Einstein-Notation for Arrow Terms

To avoid the many set-theoretical parenthesis, the following notation, called *Arrow Schemes*, is applied, in analogy to the Einstein notation [22, p. 6]:

- a_i for a finite set of arrow terms, i denoting some Choice Function selecting finitely many specific terms out of a set of arrow terms a.
- a_1 for a singleton set of arrow terms; i.e., $a_1 = \{a\}$ where a is an arrow term.
- $\emptyset$ for the empty set, such as in the arrow term $\emptyset \to a$.
- $a_i + b_j$ for the union of two observation sets a_i ,b_j.

(8)

The application rule for M and N now reads:

$$M \bullet N = (a_i \to b) \bullet N = \{b | \exists a_i \to b \in M;\ a_i \subset N\} \quad (9)$$

$(a_i \to b) \subset M$ is the subset of level 1 arrow terms in M. With these conventions, $(a_i \to b)_j$ denotes a *Concept*, i.e., a non-empty finite set of arrow terms with level 1 or higher, together with two choice functions i, j. Each set element has at least one arrow.

The choice function i chooses specific observations a_i out of a (larger) set of observations a. This is what Zhong describes as *Grounding* when linking observations to real-world objects [23]. In AI, grounding is crucial for linking AI engines to the real world. If a denotes knowledge, i.e., an infinite set of arrow terms of any level, a_i can become part of a concept consisting of specific arrow terms referring to some specific object, specified by the choice function i. Choice functions therefore have the power of focusing knowledge on specific objects in specific areas. That makes choice functions interesting for intelligent systems and AI.

There is a conjunction of choice functions, thus $a_{i,j}$ denotes the union of a finite number of grounded arrow schemes:

$$a_{i,j} = a_{i,1} \cup a_{i,2} \cup \dots \cup a_{i,m} = \bigcup_{k=1}^{m} a_{i,k} \quad (10)$$

There is also cascading of choice functions. Assume $N = \left(a_j \to b\right)_k$, then:

$$\begin{gathered} M = \left(\left(\left(a_j \to b\right)_k \to b_i\right)_l \to c\right) \text{ and} \\ M \bullet N = \left(b_{i_l} \to c\right) \end{gathered} \quad (11)$$

The choice function might be used for grounding an arrow scheme to observations.

An arrow scheme without outer indices represents a potentially infinite set of arrow terms. Thus, writing a, we mean knowledge about an observed object. Adding an index, a_j, indicates such a grounded object together with a choice function j that chooses finitely many specific observations or knowledge.

While on the first glimpse, the Einstein notation seems just another way of denoting arrow terms, for representing such data in computers it means that the simple enumeration of finite data sets is replaced by an intelligent choice function providing grounding that must be computed and can be either programmed or guessed by an intelligent system.

For practical applications, the choice function is an important part of deep learning. It means learning by generalization. The more choices you get on the left-hand side, the more knowledge you acquire. The ARC price competition for instance is easily solvable if we can generalize our choice functions good enough, to draw conclusions from the samples into general rules. However, generalization is not easily available with current AI technology. *Controlling Combinators*, see section IV.B, are a workaround.

C. The Graph Model of Combinatory Logic

The algebra of observations represented as arrow terms is a combinatory algebra and thus a model of combinatory logic. The following definitions demonstrate how the graph model implements Curry's combinators **S** and **K** fulfilling equations (1) and (2), following [5].

- $\mathbf{I} = a_1 \to a$ is the Identification, i.e., $(a_1 \to a) \bullet b = b$
- $\mathbf{K} = a_1 \to \emptyset \to a$ selects the 1st argument:
  $\mathbf{K} \bullet b \bullet c = (b_1 \to \emptyset \to b) \bullet b \bullet c = (\emptyset \to b) \bullet c = b$
- $\mathbf{KI} = \emptyset \to a_1 \to a$ selects the 2nd argument:
  $\mathbf{KI} \bullet b \bullet c = (\emptyset \to c_1 \to c) \bullet b \bullet c = (c_1 \to c) \bullet c = c$
- $\mathbf{S} = \left(a_i \to \left(b_j \to c\right)\right)_1 \to (d_k \to b)_i \to \left(a_i + b_{j,i} \to c\right)$

(12)

Therefore, the algebra of observations is a model of combinatory logic. The interested reader can find complete proofs in Engeler [1, p. 389].

The *Lambda Theorem* from Barendregt [20] says that with **S** and **K,** an abstraction operator can be constructed that adds algorithmic skills to knowledge represented as arrow schemes, following equation (4).

As the name "graph model" suggests, arrow terms are an algebraic way of describing neural networks. Thus, something that nature uses to acquire and work with knowledge.

Figure 1 illustrates the effect of the combination according to equation (9). It becomes apparent that the graph model describes graphs indeed, with loops. Repeatedly applying equation (9) leads to what we perceive as the "response of a neural network". Combination of knowledge and combinators thus play a significant role in AI.

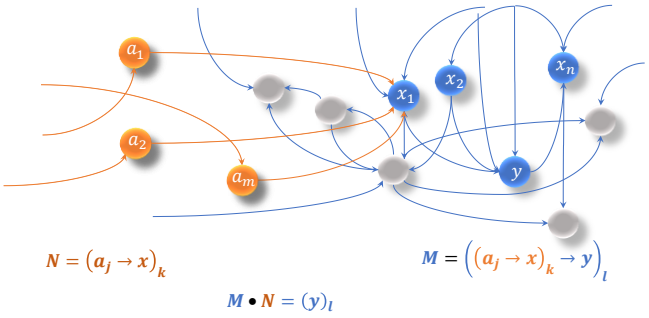

Figure 1: Neural Network become a Combinatorial Algebra

However, Figure 1 is not only a picture of an abstract graph. It can also be understood as a part of an ANN – or of an NNN. Engeler associated neuroscience with the graph model in 2019, by explaining how a brain works [24]. He used the graph model of combinatory logic as an algebraic representation of NNN.

IV. TOWARDS INTELLIGENT SYSTEMS

Barceló et. al. has shown in 2019 that modern neural network architectures are Turing-complete [25]. This is also a property of the graph model but not of every ANN. We propose an architecture for intelligent systems that incorporates conventional algorithmic programming.

A. How Arrow Schemes describe ANNs

While it is obvious how an NNN is represented by arrow schemes, this is not equally clear for ANNs. The reason is that directed graphs contain loops while looping in ANNs is very restricted. There exist certain architectures for ANNs that allow for loops, within narrow limits, a typical *Multi-Layered Perceptron* (MLP) as used for LLMs does not [12].

Consequently, an ANN has only a limited ability to emulate an NNN.

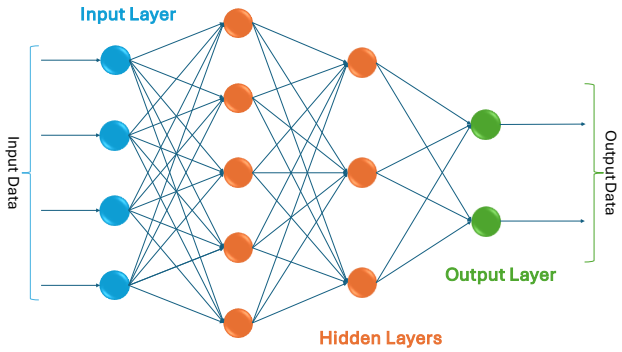

Figure 2: Multi-Layered Perceptron as an ANN

In principle, every arrow scheme $a_i \to b$ describes one node in a directed but not loop-free graph. Some arrow schemes describe algorithmic concepts such as in equation (12) or as explained in equation (5). Other arrow schemes simply connect observations a_i to some response b. General knowledge has many facets.

It would be wonderful if we had the ability to look at an LLM and identify arrow schemes for each node. This would add full explainability to AI, but unfortunately, this is not the case. Theoretically, this is because neither combinatory terms nor arrow schemes have normal forms. Very often there is a wide variety of solutions that are equivalent but widely different in effectiveness.

This makes explainability of AI difficult. The lack of normal forms blocks all attempts to find the one sequence of arrow schemes that explains what AI is doing. AI engineers have no other choice than trying to train their ANNs such that the response meets expectations but without exactly knowing what happens. It is comforting, however, that they share the same sad fate with neuroscientists. It is astonishing how long-forgotten theoretical results such as the lack of a normal form in combinatory logic yield economically highly relevant results, nowadays, in the evolving AI ecosystem. Consult Lachowski [25] for a survey of the performance challenges that occur around combinatory logic.

However, there is a famous saying that nothing is too difficult for the engineer ("Inventor of Anything"). Recent findings suggest that AI is capable of recognizing chains of thought that lead to the observation of a specific response [26]. This complements earlier findings that describe CoT as a prompting technique [27]. Thus, there exist AI architectures that allow to identify at least some arrow schemes that describe what AI does. It is not necessarily the full truth, as is

also not the case when humans explain their thoughts to colleagues. But it should be enough to persuade them.

Having a complete sequence of arrow schemes describing some ANN would lead to explainable AI that even is able to get certified for safety-critical applications. However, there is a problem with hidden layers. While the *Quality Function Deployment* (QFD) method uses identifiable topics for each layer [28], an ANN has none; they are hidden indeed. Thus, much of the intermediate reasoning also remains hidden. RQ 2 remains at least partially unanswered. If the input data and response only can be captured by arrow schemes, the intermediate steps must be guessed based on domain knowledge, but it is not known what exactly the AI engine actually did consider. AI might change behavior and create hazardous changes to the hidden layers; low-rank adaptation (LoRA) of large language models is an attempt to limit such change [29]. In QFD, on the contrary, intermediate stages are identifiable based on their topic; for example, when deploying customer needs, we first go to user stories and then to testable features.

Another approach to better explainable AI is already well established: *Retrieval-Augmented Generation* (RAG) might avoid hallucinations for LLMs [30] by referencing knowledge databases and including them into the generation of responses. RAG impacts the architecture of intelligent systems by connecting neural networks to knowledge databases [31]. RAG corresponds to grounding arrow schemes using the choice function; RAG is indispensable for explainable AI.

This is the motivation for looking at AI architectures. In some way, it must be complemented by functionality that controls the behavior of AI. Only with such control an AI-engine can perform safety-critical tasks. When certifying an AI-engine for safety, it is no longer necessary to convert all nodes of an ANN into arrow schemes, but we can focus on the overall result. If an AI fails on such tasks, we do not have a white-box trace of all nodes, or arrow schemes, that have contributed to this failure, but we are at least as good as with traditional safety-preserving methods and techniques.

B. *The Architecture of Intelligent Systems*

Intelligent systems using AI are based upon *Controlling Combinators*. Controlling combinators are derived from the idea behind fixpoint combinators, see equation (5). A *Controlling Operator* $\mathbf{C}$ acts on a controlled object X by its application $\mathbf{C} \bullet X$. Control means that knowledge represented by arrow schemes in X is completely known and described.

Accomplishing control can be formulated by (13):

$$\mathbf{C} \bullet X = X \tag{13}$$

The equation (13) is a theoretical statement, referring to a potentially infinite loop. For solving practical problems, X must be approximated by finite subterms.

Thus, the control problem is solved by a *Control Sequence* $X_0 \subseteq X_1 \subseteq X_2 \subseteq \cdots$, a series of finite subterms and the controlling operator $\mathbf{C}$, starting with an initial X_0 and determined by (14):

$$X_{i+1} = \mathbf{C} \bullet X_i, i \in \mathbb{N} \tag{14}$$

This is called *Focusing*. The details can be found in Engeler [24, p. 299]. The controlling operator $\mathbf{C}$ gathers all faculties that may help in the solution. The inclusion operator in equation (14) is explained by the graph model. The control problem is a repeated process involving substitution, like finding the fixpoint of a combinator, and thus increasing the number of arrow schemes, and especially of choice functions, in the resulting focusing process.

Controlling combinators both collect and use empirical data for continuous training. Such an intelligent system incorporates the necessary functional processes for fine-tuning based on feedback received.

For more details, we refer to the authors' paper about solving the control problem [32].

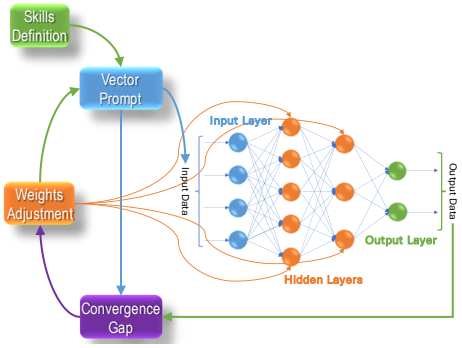

Figure 3: Self-learning Intelligent Systems based on an ANN

The program scheme we use in Figure 3 for the controlling combinator depends on the *Convergence Gap*; the measurable variation between actual behavior and expectations and requirements regarding an AI-enabled intelligent system. Both come as (large) vectors and thus the Euclidean distance is easily computable. Expectations and correct answers might also come from an external knowledge database, allowing the intelligent system to learn autonomously.

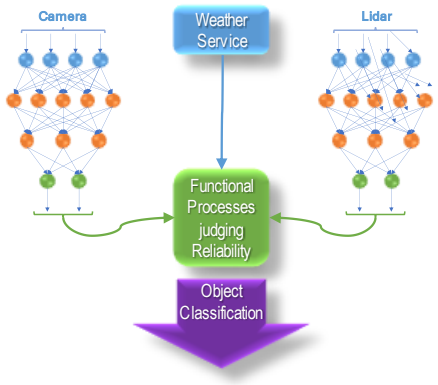

Figure 4: An Intelligent System that selects the most reliable AI response.

The architecture for RAG now extends. Instead of embedding the reference into response generation [31], and hoping it works, we set up functional processes for comparing LLM results with evidence from the knowledge database and calculate the convergence gap.

The convergence gap of such a system fully explains its behavior. Under well-defined conditions, such a system can be certified, even for safety critical tasks.

It is also possible to add more than one AI engine to an intelligent system, compare results and go forward with the most reliable one. Insufficient training, biases, and hallucinations therefore would become detectable.

Figure 4 shows an example of an intelligent system design that relies on two separate visual recognition engines analyzing the same scenario, one through a camera and the other through a Lidar. Such an architecture requires that the reliability of each artificial intelligence engine be known, under certain conditions, such as weather. In this way, the intelligent system can explain why it selected one or the other response.

Obviously, if both AI-engines produce an identical response, this increases overall reliability of the response of the intelligent system quite a bit.

The graph model delivers the metrics for defining controlling combinators by inclusion, and it also allows to combine knowledge and thus reliability correctly, by equation (9). This is discussed in another paper of the authors [33]. This remark should also explain why we do not use the term "Loss Function" that originates from *Signal Theory* and originally described the loss of fidelity in analog sound transmission. Since the discovery of the *Fast-Fourier Transform* (FFT) [34], one understands that A/D-convergence is not a loss, but an acquisition of enough knowledge to reach some threshold. Deep learning uses the same principles.

V. Conclusions and Future Work

We therefore have shown that

RQ 1: ANNs can be represented in the graph model of combinatory logic;
RQ 2: CoT do not exactly relate to a sequence of arrow schemes, as they do not cover hidden layers in ANNs;
RQ 3: Arrow schemes do not explain AI but explain how AI can be controlled.

The graph model of combinatorial logic does not provide an alternative for implementing AI, but it is an excellent guide and theoretical foundation for what can be done with AI, for explaining AI, but also for learning where AI meets its limits.

The current step forward is collecting several designs of intelligent systems, finding methods for measuring reliability and defining suitable convergence gaps. This work in progress of the authors will be shared with interested parties [35]; the authors have no institution or sponsor to help with this.

It remains the idea that AI could be explained by searching for arrow schemes that provide the same responses. Since combinatory logic does not have normal forms, this seems feasible. It could be used as a validation process for AI. However, for now, this is a future research project.

Acknowledgment

The authors would like to thank Lab42 in Davos for asking excellent questions and promoting the ARC Challenge [9], now ARC Prize [36], and to the anonymous IARIA reviewers who helped to improve this paper quite a bit. Special thanks to Erwin Engeler for his suggestions and availability for valuable discussions with his former student.

References

[1] E. Engeler, "Algebras and Combinators," *Algebra Universalis*, vol. 13, pp. 389-392, 1981.

[2] K. Gödel, "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I," *Monatshefte für Mathematik und Physik*, vol. 38, no. 1, p. 173–198, 1931.

[3] A. Turing, "On computable numbers, with an application to the Entscheidungsproblem," *Proceedings of the London Mathematical Society*, vol. 42, no. 2, pp. 230-265, 1937.

[4] P. Raatikainen, "Gödel's Incompleteness Theorems," in *The Stanford Encyclopedia of Philosophy*, E. N. Zalta, Ed., 2020.

[5] H. Curry and R. Feys, Combinatory Logic, Vol. I, Amsterdam: North-Holland, 1958.

[6] A. Church, "The Calculi Of Lambda Conversion," *Annals Of Mathematical Studies 6*, 1941.

[7] F. Rosenblatt, "The Perceptron: A Perceiving and Recognizing Automaton (Project PARA)," Cornell Aeronautical Laboratory, Inc., Buffalo, 1957.

[8] M. Minsky and S. Papert, Perceptrons: An Introduction to Computational Geometry, 2nd edition with corrections ed., Cambridge, MA: The MIT Press, 1972.

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

[10] K. Wijekumar, B. J. Meyer and P. Lei, "High-fidelity implementation of web-based intelligent tutoring system improves fourth and fifth graders content area reading comprehension," *Computers & Education*, vol. 68, pp. 366-379, 2013.

[11] A. Peterson, "Literacy is More than Just Reading and Writing," National Council of Teachers of English, 23 March 2020. [Online]. Available: https://ncte.org/blog/2020/03/literacy-just-reading-writing/. [Accessed 4 July 2024].

[12] M. v. Gerven and S. Bohte, "Artificial Neural Networks as Models of Neural Information Processing," in *Frontiers in Computational Neuroscience*, Lausanne, 2017.

[13] IBM TechXchange Community, "What is explainable AI?," IBM Reserach, [Online]. Available: https://www.ibm.com/topics/explainable-ai. [Accessed 23 10 2024].

[14] F. Dallanoce, "Explainable AI: A Comprehensive Review of the Main Methods," Medium.com, San Francisco, California, 2022.

[15] M. R. Morris, J. Sohl-Dickstein, N. Fiedel, T. Warkentin and A. Dafoe, "Levels of AGI for Operationalizing Progress on the Path to AGI," arXiv:2311.02462v4 [cs.AI], Cornell University, 2024.

[16] H. Curry, J. Hindley and J. Seldin, Combinatory Logic, Vol. II, Amsterdam: North-Holland, 1972.

[17] T. M. Fehlmann and E. Kranich, "Intuitionism and Computer Science – Why Computer Scientists do not Like the Axiom of Choice," *Athens Journal of Sciences*, vol. 7, no. 3, pp. 143-158, 2020.

[18] T. M. Fehlmann and E. Kranich, "A General Model for Representing Knowledge - Intelligent Systems Using Concepts," *Athens Journal of Sciences*, vol. 11, pp. 1-18, 2024.

[19] H. Barendregt and E. Barendsen, Introduction to Lambda Calculus, Nijmegen: University Nijmegen, 2000.

[20] H. P. Barendregt, "The Type-Free Lambda-Calculus," in *Handbook of Math. Logic*, vol. 90, J. Barwise, Ed., Amsterdam, North Holland, 1977, pp. 1091 -1132.

[21] T. M. Fehlmann and E. Kranich, "The Fixpoint Combinator in Combinatory Logic - A Step towards Autonomous Real-time Testing

of Software?," *Athens Journal of Sciences*, vol. 9, no. 1, pp. 47-64, 2022.

[22] T. M. Fehlmann, Autonomous Real-time Testing – Testing Artificial Intelligence and Other Complex Systems, Berlin, Germany: Logos Press, 2020.

[23] V. Zhong, J. Mu, L. Zettlemoyer, E. Grefenstette and T. Rocktäschel, "Improving Policy Learning via Language Dynamics Distillation," arXiv:2210.00066v1, Cornell University, 2022.

[24] E. Engeler, "Neural algebra on "how does the brain think?"," *Theoretical Computer Science*, vol. 777, pp. 296-307, 2019.

[25] Ł. Lachowski, "On the Complexity of the Standard Translation of Lambda Calculus into Combinatory Logic," *Reports on Mathematical Logic*, vol. 53, pp. 19-42, 2018.

[26] Z. Jin and W. Lu, "Self-Harmonized Chain of Thought," arXiv:2409.04057v1 [cs.CL], Cornell University, 2024.

[27] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le and D. Zhou, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," arXiv:2201.11903v6 [cs.CL], Cornell University, 2023.

[28] T. M. Fehlmann and E. Kranich, "How to Explain Artificial Intelligence to Humans - Learning from Quality Function Deployment," in *Systems, Software and Services Process Improvement - CCIS 2179*, vol. Part 1, Murat Yilmaz, Paul Clarke, Andreas Riel, Richard Messnarz, Christian Greiner and Thomas Peisl, Eds., Munich, Springer Communications in Computer and Information Science 2179, EuroSPI 2024, pp. 48-63.

[29] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang and W. Chen, "LoRA: Low-Rank Adaptation of Large Language Models," arXiv:2106.09685v2 [cs.CL], Cornell University, Ithaca, NY, 2021.

[30] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang and H. Wang, "Retrieval-Augmented Generation for Large Language Models: A Survey," arXiv:2312.10997v5 [cs.CL] , Cornell University, 2024.

[31] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel and D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," arXiv:2005.11401v4 [cs.CL], Cornell University, 2021.

[32] T. M. Fehlmann and E. Kranich, "Making Artificial Intelligence Intelligent - Solving the Control Problem for Artificial Neural Networks by Empirical Methods," in *Human Systems Engineering and Design (IHSED2024)*, Split, Croatia, 2024.

[33] T. M. Fehlmann and E. Kranich, "Measuring Knowledge - An Attempt to Define a Measurement Principle for Learning Machines," *ATINER Journal of Science*, 2024.

[34] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Mathematics of Computation*, vol. 19, pp. 297-301, 17 August 1964.

[35] T. M. Fehlmann, "Intelligent Systems," Euro Project Office AG, 9 May 2024. [Online]. Available: https://web.tresorit.com/l/AXX78#FaBkGqfY2cF_JsVmX70_ng.

[36] Lab 42, "ARC Price," Mike Knoop, Davos{San Francisco, 2024.