Intelligent Human Systems Integration (IHSI 2024), Vol. 119, 2024, 120–126
https://doi.org/10.54941/ahfe1004475

The Neural Algebra and Its Impact on Design and Test of Intelligent Systems

Thomas Fehlmann[1] and Eberhard Kranich[2]

[1]Euro Project Office, 8049 Zürich, Switzerland
[2]Euro Project Office, 47051 Dortmund, Germany

ABSTRACT

The Graph Model of Combinatory Logic (Engeler, 1981) is also a mathematical model for "how does the brain think". It attempts to explain how complex scripts of behaviour and conceptual content can reside in, combine, and interact on large neural networks (Engeler, 2019). This has an impact on building intelligent systems that interact with humans. Intelligent systems should employ the same kind of concepts humans do; otherwise, their actions remain incomprehensible and erratic to human users, concepts can be represented in the Graph Model by using the "Lambda-Theorem" found by Barendregt (Barendregt, 1977). Both the Graph Model and the mathematical model for the human brain have been published as part of Theoretical Computer Science and remain thus out of the reach of normal AI engineers. Nevertheless, they suggest solutions for today's problems with Intelligent Systems, such as autonomous vehicles mastering the traffic in Palermo, or robots caring for people and working together with them. Intelligent Systems need to adhere to concepts quite like humans that follow certain rules in their behaviour. The paper explains what a "Concept" is in AI, how to state requirements for AI, and how to test them. Intelligent Systems using concepts behave like humans, following rules but are still able to break the rules when need arises, and can be certified for safety and security; solving certain difficulties for Learning Machines that can learn and unlearn.

Keywords: Combinatorial logic, The graph model, Neural algebra, Artificial intelligence, Learning machines, Intelligent systems

INTRODUCTION

Intelligent systems powered by tools of Artificial Intelligence (AI) such as neural networks (NN), deep learning (DL), support vector machines (SVM), large language models (LLM) and generative pretrained translators (GPT) have a disturbing impact on humans: There are unreliable (Gunn, 1998). And their responses remain unexplainable (Phillips, et al., 2021).

Sensors do most often, but not always, recognize signals. Image processing can often, but not always, distinguish wolves from dogs, or chihuahua from mandarins. This makes them suspect to humans who are accustomed to trust others based on a social consent. We trust a taxi driver because he is licensed, and we consider someone as knowledgeable because other do as well, or because he or she has a degree as master or doctor. Who guarantees me that responses from ChatGPT are reliable, and correct? How can I ever be sure that an autonomous car taxi will safely take me to my destination?

AI can not only learn; it also can unlearn. If trained by a human, a neural network will effectively copy its biases and, in case, any ruthless behaviour it exposes. This is the motivation why we should spend some effort in understanding what knowledge is (Fehlmann & Kranich, to appear, 2023).

The Graph Model of Combinatory Logic as a Model for Knowledge

Schönfinkel and Curry developed *Combinatory Logic* (Curry & Feys, 1958) to avoid the problems introduced when using logical quantifiers, and Church invented *Lambda Calculus* as a rival but equivalent formalism (Church, 1941). Combinatory logic is an algebra with just one operation, the application of one combinator to another, without considering anything resembling typed terms. With combinatory logic, one had a foundation for mathematics that worked without quantifiers and thus avoided the undecidability problem for predicate logic, found by Gödel (Gödel, 1931).

We start with a non-empty set $\mathcal{L}_0$, represented as nodes in Figure 1. $\mathcal{L}_0$ constitutes the base of our recursively defined arrow terms. Let a_i be a finite set of arrow terms, and b another arrow term. Then, all terms of the form (1) are arrow terms:

$$a_i \to b \tag{1}$$

The name originates from the intended meaning of neurons a_i firing towards a common destination b. Arrow terms represent a directed graph where a_i depict the vertices of origin and b the destination vertex. Engeler has shown that the recursively defined algebra of arrow terms sets constitutes a model of combinatory logic (Engeler, 1981), because it contains an application operation (2) that gives the graph model an algebraic structure modelling application in combinatory logic.

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

The role of the index i in a_i is that of a choice function that selects a finite subset of some observation class, namely the set of arrow terms a.

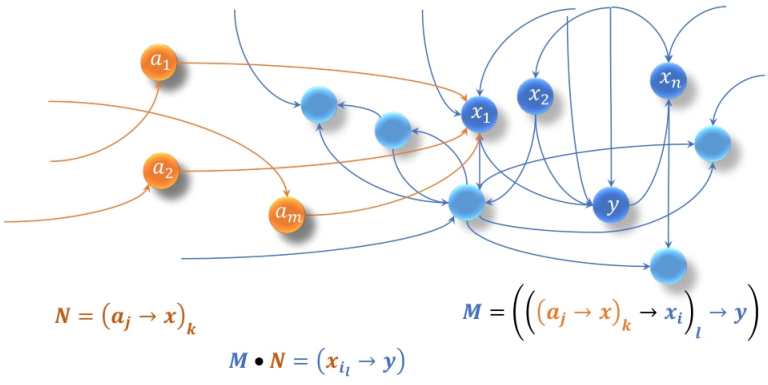

Figure 1: The graph model of combinatory logic – visualization with application $M \bullet N$.

If $\mathcal{L}_0, \mathcal{L}_1, \mathcal{L}_2, \ldots$ denote the nesting level of arrow terms, the union of all levels $\bigcup \mathcal{L}_i$ is closed under application operation (2). Arrow terms without a single arrow are elements of $\mathcal{L}_0$ and are called *Observations*. Arrow terms that are not observations are called *Concepts*. They contain one arrow term level at least. The recursivity of the definition of the arrow term elements $\bigcup \mathcal{L}_i$ becomes quite natural with the visualization in Figure 1. It simply means that the graph is infinite and not bounded by anything like basic observations. An observation can always be represented by any single node in the graph; a concept by two adjacent neurons connected by a "firing" arrow.

Note that neurophysiologists will have a hard time identifying something that is not a concept in our brain. The idea that sensor cells are the $\mathcal{L}_0$ elements in our brain is a logical construct that is not supported by real physical observations. In intelligent systems, we might consider observations to be the only output of a sensor. However, sensors can also consist of a different neural network of their own. In the human body, we find neural structures that respond to external stimuli such as light, heat, or motion, but these are rarely single cells.

The graph model also includes lambda terms (Barendregt & Barendsen, 2000). *Lambda Terms* have the form λx.M where x represents a "variable" in M, allowing for an application of the Lambda term λx.M on some argument N

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

In this case (3), N replaces all occurrences of x in M. For formal definitions, consult (Fehlmann, 2020, p. 5).

The graph model of combinatory logic can be used to explain how the human brain works (Engeler, 2019). In fact, the directed graphs represented by arrow terms can be interpreted as a neurological model representing neurons firing on others and causing a reaction.

Barendregt's Lambda calculus (3) 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 rules exist like those used in social interactions between humans.

Requirements for Intelligent Systems

When designing intelligent systems, it is not sufficient to specify the objects that it must be able to recognize, but also the concepts that it must be able to learn and especially those that have a compulsory nature and must be implemented using Lambda concepts.

Thus, requirements are classified in three categories that correspond to three different technical solutions. The first category entails the objects an intelligent system should be able to recognize for physical interaction. It is assumed that the system uses sensors, such as cameras, touch sensors, and microphones, to find its way in its surroundings. The second category are concepts that the system learns, using machine learning and by training neural networks. Both, objects, and trained concepts, cannot be determined with certainty; there is always a small error range where the system fails. This

might occur because sensors are not accurate enough, or by malfunction, or because training always is limited, leaving some uncertainty and gaps. Such requirements have a significant degree of indeterminacy that in turn is more difficult to validate.

Compulsory concepts, in contrary, can be identified by *Functional User Requirements* (FUR) as for traditional programs, defining deterministic functionality and with the possibility to validate their implementation under controlled conditions, like testing software.

Table 1 shows the three categories of requirements that should be clearly identified and distinguished in the requirements elicitation process. Traditional functional requirements are always in category three, as there is no level of uncertainty in implementation allowed. Strictly speaking, functional tests can only be performed for category 3 requirements; however, the other categories shall be subject to *Autonomous Real-time Testing*, as explained in a forthcoming chapter, and the book (Fehlmann, 2020).

Table 1. Requirements for intelligent systems.

Type	Requirement	Technical Solution
❶	What objects should it recognize? • Classical Artificial Intelligence • Pattern Recognition • Tagging scenarios & objects	Training Models • Samples from a typical world • Deep Learning, continuous updates • Neural networks
❷	What concepts should it learn? • Some concepts are easier to learn than being programmed. • Continuously adapting to user's behaviour and preferences	Learning Concepts • Learning to perform actions that are typical in its environment. • Continuously adapting by collecting and evaluating experiences
❸	Which concepts are compulsory? • Safety • Security • Legal rules	Lambda Concepts • Reacting on specific scenarios • Predefining behaviour • Compulsory decisions

How to Make Artificial Intelligence Reliable

Technically, based on the requirement types in Table 1, it is straightforward:

- Build intelligent systems according to requirements that contain type ❶, type ❷, and type ❸ requirements. Use the type ❸ requirements to let the intelligent system stick to certain rules and behaviour, thus looking reliable to humans,
- Test the intelligent system accordingly:
    - Type ❶ requirements cannot be fulfilled at 100% but sensors, neural networks and whatever else can produce observations of the physical environment have a certain reliability level;
    - Type ❷ requirements address concepts that must be learned, be it supervised or unsupervised, and they too might still contain a failure

rate that exceeds 0%. Thus, their reliability is limited to something below 100%;
- Type ❸ requirements, in turn, address conventional programming. Programmers implement rules as Lambda concepts in an intelligent system that behaves predictably like a traditional machine.

With such a strategy, intelligent systems behave predictably and thus create trust among humans.

Socially it is less straightforward. While humans pass exams to get certified or licensed, with intelligent systems this is less easy – although for certain intelligent systems such as medical instruments or vehicles for daily traffic, the processes exist. But the existing processes do not yet address the specific challenges by systems that learn and change behaviour while in operation.

However, unless testing has been adapted to the needs of intelligent system, the social aspect is not covered. People must trust intelligent systems before they hand oneself over to such systems. You cannot force trust; you must design the system to enable trust.

How to Test Intelligent Systems

Testing of intelligent systems is not something that can be done before release; it must occur anytime, continuously, during its operational use or while not in use. Moreover, users must be notified about the nature and outcome of tests. Only then people will board an autonomous car, or taxi, with the same confidence that they exhibit against human drivers.

Intelligent systems learn and change behaviours. Testing before release is not an option, because after release the responses to certain incidences will change. Intelligent systems adjust to their environment, or to their users and owners.

Today's software testing practices are way behind the age of digitalization. No generally adopted metrics exist for test intensity that can be applied to compare competing intelligent systems. It remains unclear whether the entire functionality required to control intelligent systems is tested at all, or only the part of the system that is executed by self-written code. Because software changes continuously, with *Continuous Integration/Continuous Delivery*, tests executed at release quality gates only reflect the original state of delivery, in isolated environments. Intelligent cyber-physical systems affect the real world, unfortunately, and they do continuously. *Continuous Testing* must therefore cover the entire life cycle, including operations.

For this reason, we proposed *Autonomous Real-time Testing* (Fehlmann, 2020). It means that:

- Tests are model-based to cover all functionality, including services from the cloud or from AI-components;
- Tests are measured by functional size according to ISO/IEC 19761 (ISO/IEC 19761, 2019) for comparing test size with functional size;
- Test suites are continuously expanded by new test cases, applying AI methods to meet user's needs;

- Tests are fully automated and can occur autonomously when the product is not in use;
- Tests of Lambda concepts are included and made explicit;
- The user of the intelligent systems gets informed about the status and results of continuous testing while operating the intelligent system.

Autonomous Real-time Testing is not yet available; technology is mostly there but its implementation is lagging. AI-based tools can automate testing by creating suitable test cases and executing them. But this is not enough. Users must know about the status of tests, and understand what is being tested, otherwise they will probably never trust something such as an autonomous car.

CONCLUSION

Human machine communication becomes a lot more complicated with the advent of AI in daily products. The experiences made with ChatGPT have taught us that the technical possibilities produce much scepticism even among promoters of the technical progress. Indeed, the fact that ChatGPT uses Wolfram Research's *Wolfram|Alpha* (Wolfram, 2023) as a source for knowledge about mathematics, physics and natural science shows that adding Lambda concepts to knowledge is nothing unexpected, or strange. Even if promoters of AI stress the potential of learning concepts, only Lambda concepts can ensure safety standards and create trust as needed to make intelligent systems successful as consumer products.

It's all a matter of human-machine interaction that must be properly addressed.

ACKNOWLEDGMENT

The authors would like to acknowledge Lab42 in Davos, Switzerland, for helping to understand the need for Lambda concepts in AI.

REFERENCES

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

Barendregt, H. P., 1977. The Type-Free Lambda-Calculus. In: J. Barwise, Hrsg. *Handbook of Math. Logic*. Amsterdam: North Holland, pp. 1091–1132.

Church, A., 1941. The Calculi of Lambda Conversion. *Annals of Mathematical Studies 6*.

Curry, H. & Feys, R., 1958. *Combinatory Logic, Vol. I*. Amsterdam: North-Holland.

Engeler, E., 1981. Algebras and Combinators. *Algebra Universalis,* Band 13, pp. 389–392.

Engeler, E., 2019. Neural algebra on "How does the brain think?". *Theoretical Computer Science,* Band 777, pp. 296–307.

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

Fehlmann, T. M. & Kranich, E., to appear, 2023. A General Model for Representing Knowledge - Intelligent Systems Using Concepts. *Athens Journal of Sciences*.

Gödel, K., 1931. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für Mathematik und Physik*, 38(1), pp. 173–198.

Gunn, S., 1998. *Support Vector Machines for Classification and Regression*, Southampton: ISIS Technical Report, University of Southampton.

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

Phillips, P. J. et al., 2021. *Four Principles of Explainable Artificial Intelligence*, Washington DC, USA: NIST - National Institute of Standards and Technology.

Wolfram, S., 2023. *What is ChatGPT doing ... and Why Does it Work?*. Champaign, IL: Wolfram Media, Inc.