AGI via Combining Logic with Deep Learning

甄景贤 (King-Yin Yan)

General.Intelligence@Gmail.com

Abstract. An integration of deep learning and symbolic logic is proposed, based on the Curry-Howard isomorphism and categorical logic. The propositional structure of logic is seen as a symmetry, namely the permutation invariance of propositions; This can be implemented using so-called symmetric neural networks. Under our interpretation, it turns out that Google's BERT, which many currently state-of-the-art language models are derived from, can be regarded as an alternative form of logic. This BERT-like structure can be incorporated under a reinforcement-learning framework to form a minimal AGI architecture. We also mention some insights gleaned from category and topos theory that point to future directions and may be helpful to other researchers, including mathematicians interested in AGI.

Keywords: deep learning, symbolic logic, logic-based AI, neural-symbolic integration, Curry-Howard isomorphism, category theory, topos theory, fuzzy logic

0 Introduction

Results in the present paper does not make use of category theory in any significant way (nor the Curry-Howard isomorphism, for that matter). Its main accomplishment is to express AGI in the categorical language. To the lay person, concepts of category theory (such as pullbacks, adjunctions, fibration, toposes, sheaves, ...) may be difficult to grasp, but they are the mathematician's "daily bread". We hope that describing AGI in categorical terms will entice more mathematicians to work on this important topic.

Secondly, an abstract formulation allows us to see clearly what is meant by "the mathematical structure of logic", without which logic is just a collection of esoteric rules and axioms, leaving us with a feeling that something may be "amiss" in our theory.

0.1 The Curry-Howard Isomorphism

As the risk of sounding too elementary, we would go over some basic background knowledge, that may help those readers unfamiliar with this area of mathematics.

The Curry-Howard isomorphism expresses a connection between logic **syntax** and its underlying **proof** mechanism. It is fundamental to understanding categorical logic. Consider the mathematical declaration of a **function** f with its domain and co-domain:

$$f : A \to B. \tag{1}$$

This notation comes from type theory, where A and B are **types** (which we can think of as sets or general spaces) and the function f is an **element** in the function space $A \to B$, which is also a type.

What the Curry-Howard isomorphism says in essence is that we can regard $A \to B$ as a **logic** formula, ie. the implication $A \Rightarrow B$, and the function f as a **proof** process that maps a proof of A to a proof of B. [1]

The following may give a clearer picture:

$$\begin{array}{ll} \boxed{\text{logic}} & A \Longrightarrow B \\ & \text{- - - - - - - - - -} \\ \boxed{\text{program}} & \blacksquare \overset{f}{\longmapsto} \blacksquare \quad . \end{array} \tag{2}$$

What we see here is a logic formula "on the surface", with an underlying proof mechanism which is a **function**, or λ-calculus term. Here the $\blacksquare$'s represent proof objects or **witnesses**. The logic propositions A and B coincide with the **domains** (or **types**) specified by type theory. Hence the great educator Philip Wadler calls it "propositions as types". [2] Other textbooks on the Curry-Howard isomorphism include: [40] [39] [42].

The gist of our theory is that Deep Learning provides us with neural networks (ie. non-linear functions) that serve as the proof mechanism of logic via the Curry-Howard isomorphism. With this interpretation, we can impose the mathematical structure of logic (eg. symmetries) onto neural networks. Such constraints serve as **inductive bias** that can accelerate learning, according to the celebrated "No Free Lunch" theory [45] [1] [36].

In particular, logic propositions in a conjunction (such as $A \wedge B$) are commutative, ie. invariant under permutations, which is a "symmetry" of logic and perhaps the most important one. This symmetry decomposes a logic "state"

[1] Though one does not need to execute a function to prove a statement; Merely the existence of a such a function (proof object) that type-checks is sufficient.

[2] See his introductory video: https://www.youtube.com/watch?v=IOiZatlZtGU .

into a set of propositions, and seems to be a fundamental feature of most logics known to humans. Imposing this symmetry on neural networks gives rise to symmetric neural networks (see §3).

We have not been clear about what the **proof witnesses** are. In our current implementation, types are regions in vector space and witnesses are just points inside the regions. When some propositions imply another proposition, there is a function mapping witnesses in some regions to a new witness in another region. Thus, such spatial regions are nearly tautologous with proof witnesses (ie. points versus the regions containing them). In other words, the "big" vector space is divided into many small regions representing various propositions.

We should point out that the Curry-Howard isomorphism has not played a significant role in our current AGI theory. The representation of **conditional** statements (eg. $A \Rightarrow B$) requires **function types** which are hard to represent as vectors. [3] So the only function type in our system is the "main" neural network simulating the $\vdash$ operator. In the language of classical logic-based AI, this is similar to having "Horn form" logic rules in the knowledge base, while the working memory contains *atomic* propositions only.

As an aside, the Curry-Howard isomorphism also establishes connections to diverse disciplines. Whenever there is a space of elements and some operations over them, there is a chance that it has an underlying "logic" to it (see eg. Baez and Stay's "Rosetta Stone" paper: [2], also [14]). For example, in quantum mechanics, that of Hilbert space and Hermitian operators. Another example: in String Theory, strings and cobordisms between them. For example the famous "pair of pants" cobordism (Fig. 1A), representing a process in time that merges two strings into one (time is read upwards).

Seeing logical types as topological spaces is also the origin of Voevodsky's **Homotopy Type Theory** (HoTT) [29], where the **identity** of two inhabitants in a type is seen as a homotopy **path**. HoTT may be relevant to AGI if we want the convenience of having multiple identical proofs of the same propositions – this may help simplify the topology of types (ie. spatial regions representing propositions). For example in Fig. 2A, two disjoint regions can be connected by a path, even though x_1 and x_2 are "identical" points.

[3] Ben Goertzel's latest "general theory of AGI" [11] addresses higher-order networks, which construct other networks as proofs of implications.

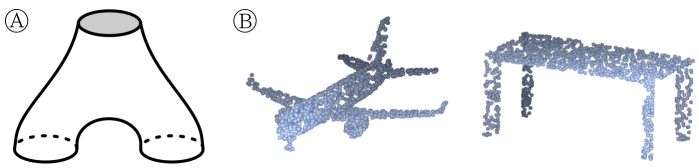


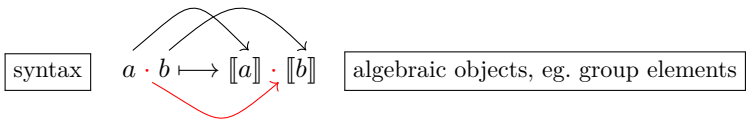
Fig. 1. Ⓐ pair of pants. Ⓑ point clouds

1 Prior Research

1.1 Neuro-Symbolic Integration

There has been a long history of attempts to integrate symbolic logic with neural processing, with pioneers such as Ron Sun, Dov Gabbay, Barbara Hammer, among others. We describe two **model-based** approaches below.

From a categorical perspective, model theory is a **functor** mapping logic syntax to algebraic objects *and* the operations between them (hence the name "functorial semantics"):



(3)

Model theory is interesting when the target structure has additional properties beyond those specified by the logic syntax. For example, the predicate `male(x)` may be modelled by:

algebraic geometry	`male`(x) $\Leftrightarrow f(x) \geq 0$	f is a polynomial
linear algebra	`male`(x) $\Leftrightarrow Mx \geq 0$	M is a matrix
topology	`male`(x) $\Leftrightarrow x \in S$	S is an open set

(4)

Model-based methods may appear impractical for AGI because the number of grounded atomic propositions gets too large (potentially infinite, if we include also propositions that are imagined). However, if all possible atoms are embedded in a mathematical space through mapping schemes such as the above (4), it may be approximately feasible.

In the "**syntactic**" or type-theoretic approach (including the one in this paper), propositions (= types) are regions in some vector space. Currently our simple scheme is to map predicates like $P(a, b)$ into the Cartesian product $\mathbb{P}\text{red} \times \mathbb{O}\text{bj} \times \mathbb{O}\text{bj}$ where $\mathbb{P}\text{red}$ is the space of all possible predicates and $\mathbb{O}\text{bj}$ the space of all possible objects [4] (but this is not the only option; see §2.1). **Inference** is

[4] Here objects mean logical or first-order objects, not categorical objects

performed by a neural network simulating the single-step consequence operator $\vdash$, while **learning** is through changing of network weights. This is relatively simple and straightforward.

Whereas, in the **model-theoretic** approach one places objects in a high-dimensional space such that their positions satisfy the constraints imposed by various predicates (eg. polynomials, matrices, open sets, ...) Now forward **inference** occurs as the system pays *attention* to (ie. to be simply aware of) some points in an $\mathbb{O}$bject space, which points are covered by some predicates. Thus a new proposition is discovered, adding to more new conclusions, ... and so on. It is interesting that, under this scheme, it seems as if all truths are known *a priori*, and the system just needs to discover or "attend" to them. **Learning** changes the geometric shapes of predicates and forms new truths to be discovered by the system.

1. In Pascal Hitzler and Anthony Seda's **Core Method** [16], an **interpretation** $\mathcal{I}$ is a function that assigns truth values to the set of all possible ground atoms in a logic language $\mathcal{L}$. One can see $\mathcal{I}$ as an enumeration of ground atoms that are true, and thus it provides a model to interpret any logic formula in $\mathcal{L}$. Moreover $\mathcal{I}$ is a function from the space X of atoms to $\mathbf{2} = \{\top, \bot\}$ and can be given a topology $\mathbf{2}^X$ which is X copies of the discrete topology of $\mathbf{2}$. Such a topology makes $\mathcal{I}$ homeomorphic to the **Cantor set** in $[0, 1]$. To a logic program P is associated a **semantic operator** $\mathcal{T}_P : \mathcal{I} \to \mathcal{I}$, performing a single step of forward **inference**. Finally, the space of interpretations $\mathcal{I}$ is embedded into $\mathbb{R}$ using a "level mapping" (The level of an atom increases by each inference step; All the atoms of an interpretation $\mathcal{I}$ are translated into a fractional number in base b). This allows $\mathcal{T}_P$ to be approximated by a neural network $f : \mathcal{I} \to \mathbb{R}$.
   The goal of their research is to find the fixed-point semantics of logic programs, but with suitable modifications, the same mathematical structure may be used to build an inference engine or AGI. In such case, the logic program would function as the **knowledge base** while interpretations would play the role of **working memory** (though the memory could only be a subset of an interpretation, due to physical limitation).
2. **∂-ILP** [9] is focused on the learning problem, but its set-up seems similar to the first example. A **valuation** is a vector $[0, 1]^n$ mapping every ground atom to a real number $\in [0, 1]$. Each clause is attached with a Boolean flag to indicate whether it is included in the results or not. From each clause c one can generate a function $\mathcal{F}_c$ on valuations that implements a single step of forward **inference**. To enable differentiability, the Boolean flag is relaxed to be a continuous value and gradient descent is used to **learn** which clauses should be included.

We would also like to mention Geoffrey Hinton's recent **GLOM theory** [15], which addresses the problem of representing a hierarchy of visual structures.

OpenCog has also been applied to neural-symbolic integration [12] [28]. These further support that representing and learning **relational** (logical) knowledge is a topic of central importance, and that there is a convergence of "mainstream" AI with AGI.

1.2 Cognitive Architectures and Reinforcement Learning

When we mention "AGI" here, it is intended to focus on a minimal core subset of its requirements, namely the ability to make logically correct inferences based on distilled knowledge learned from massive world-data. The strategy is that other modules of an AGI may be built upon this base.

Reinforcement Learning (RL). In the 1980's, Richard Sutton [41] introduced reinforcement learning as an AI paradigm, drawing inspiration from Control Theory and Dynamic Programming. In retrospect, RL already has sufficient generality to be considered an AGI theory, or at least as a top-level framework for describing AGI architectures [5].

Relation to AIXI. AIXI is an abstract AGI model introduced by Marcus Hutter in 2000 [19]. AIXI's environmental setting is the external "world" as observed by some sensors. The agent's internal model is a universal Turing machine (UTM), and the (approximately) optimal action is chosen by maximizing potential rewards over all programs of the UTM. In our (minimal) model, the UTM is *constrained* to be a neural network, where the NN's **state** is analogous to the UTM's **tape**, and the optimal weights (program) are found via Bellman optimality.

Relation to Quantum mechanics and Path Integrals. At the core of RL is the Bellman equation, which governs the update of the utility function to reach its optimal value. This equation (in discrete time) is equivalent to the Hamilton-Jacobi equation in differential form. Nowadays they are unified as the Hamilton-Jacobi-Bellman equation, under the name "optimal control theory" [24]. In turn, the Hamilton-Jacobi equation is closely related to the Schrödinger equation in quantum mechanics:

$$\boxed{\text{Bellman eqn.}} \;\leftrightsquigarrow\; \boxed{\text{Hamilton-Jacobi eqn.}} \;\leftrightsquigarrow\; \boxed{\text{Schrödinger eqn.}} \tag{5}$$

but the second link is merely "heuristic"; it is the well-studied "quantization" process whose meaning remains mysterious to this day. Nevertheless, the **path**

[5] Indeed, Sutton argues that merely increasing brute-force computing power would lead to AGI and that human design of algorithms is relatively useless. The tenet in this paper is that logic may serve as an inductive bias to accelerate learning, but we cannot be certain about this, since the algorithmic search for AGI is non-exhaustive (see §4).

integral method introduced by Richard Feynmann can be applied to RL algorithms, eg. [22].

The Hamilton-Jacobi equation gives the RL setting a "symplectic" structure [26]; Such problems are best solved by so-called symplectic integrators (proposed by 冯康 (Feng Kang) in the 1980s [10], see also [23]). Surprisingly, in the RL / AI literature, which has witnessed tremendous growth in recent years, there is scarcely any mention of the Hamilton-Jacobi structure, while the most efficient heuristics (such as policy gradient, experience replay, Actor-Critic, etc.) seem to exploit other structural characteristics of "the world".

2 The Mathematical Structure of Logic

Currently, the most mathematically advanced and satisfactory description of logic seems to base on category theory, known as categorial logic and topos theory. This direction was pioneered by William Lawvere in the 1950-60's. The body of work in this field is quite vast, but we shall briefly mention some points that are relevant to AGI. A more detailed tutorial on categorical logic, with a focus on AGI, is in preparation [46].

2.1 Predicates and Dependent Type Theory

The Curry-Howard isomorphism identifies *propositional* intuitionistic logic with type theory. As such, the arrow $\to$ in type theory is "used up" (it corresponds to the implication arrow $\Rightarrow$ in intuitionistic logic). However, predicates are also a kind of functions (arrows), so how could we accomodate predicates in type theory such that Curry-Howard continues to hold? This is the idea behind Martin Löf's **dependent type theory**.

In dependent type theory, a predicate $P(\cdot)$ is a **type constructor** ([40] §8.7) taking an element a of one type to create a new type $P(a)$. For example, each element $a \in \{\text{John, Socrates, Kermit}\}$ creates a new type Human(a), and thus Human is a **family** of types or a **dependent type**. This is depicted in Fig. 2B.

Mathematically, a dependent type is a product of types **indexed** by another type, denoted $\Pi_A B$, which is really a form of **exponentiation**. If every source element maps to the same type B, then $\Pi_A B$ *degenerates* into the ordinary function type $A \to B$ (cf. [27] §3.3).

So far, we did not make use of dependent types: predicates are represented using simple Cartesian products (ie. vector concatenation) such as $\mathbb{P}\text{red} \times \mathbb{O}\text{bj}$, but there is the possibility of exploiting more general indexing schemes.

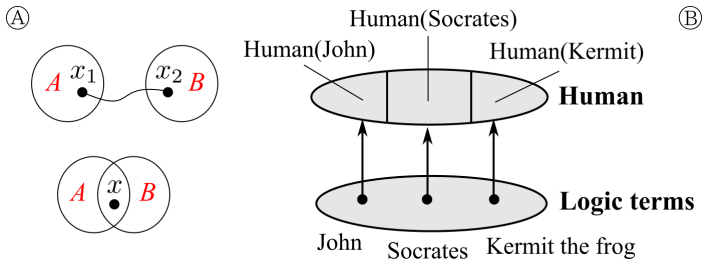


Fig. 2. Ⓐ A path in homotopy type theory. Ⓑ The predicate "Human" as a fibration

The expressiveness of predicate logic (in one form or another) is a highly desirable feature for AGI knowledge representations. So it seems necessary to incorporate dependent type theory into our logic. From a categorical perspective, predicates can be regarded as **fibers** over a base set. Fibrations capture the structure of **indexing** and **substitutions**, as shown in Fig. 2B. This figure is key to understanding Bart Jacob's book [20]. Thus category theory gives us more insight into the (predicate) structure of logic, though it is as yet unclear how to make use of this particular idea.

2.2 (Fuzzy) Topos Theory

The author's previous paper [47], almost a decade ago, proposed a fuzzy-probabilistic logic where probabilities are distributed over fuzzy truth values. So far we still believe that regarding fuzziness as a generalization of binary truth is philosophically sound. Thus it behooves to develop a generalization of standard topos theory to the fuzzy case.

A topos is a category that generalizes set theory. The most important commutative diagram in Topos theory is this one:

$$\begin{array}{ccc} X & \xrightarrow{!} & 1 \\ {\scriptstyle m}\downarrow & & \downarrow{\scriptstyle \text{true}} \\ Y & \xrightarrow[\chi_m]{} & \Omega \end{array} \tag{6}$$

It can be understood as saying that every **set** is a **pullback** of the true map $1 \to \Omega$ (which "picks out" true from $\Omega = \{\top, \bot\}$), in analogy to the idea of a "moduli space" where every family is a pullback of a "universal family" [35] [13]. Following this idea, could it be that every fuzzy set is the pullback of a fuzzy "true" map?

The book [7] §5.2.4 provides a concise review of the categorical treatment of fuzzy sets: The sub-object classifier Ω that characterizes classical set theory is

generalized to a **complete Heyting algebra** (CHA, also called a **frame**, which captures the structure of a topology, ie, the lattice of open subsets of a set; This includes the interval $[0, 1]$ as a special case, in accord with our philosophical intuition), and also leads to the recognition that *the internal logic of a topos is intuitionistic* (see [25], and this will be further explained in the tutorial [46]).

This line of research leads to Höhle's [17] and [18], where fuzzy set theory is interpreted as sub-fields of **sheave** theory, ie, complete Ω-valued sets, where Ω is a frame. More recent papers seem to be in favor of this thinking: [21] [44].

3 Permutation Symmetry and Symmetric Neural Networks

From the categorical perspective, we make the following correspondence with logic and type theory:

$$\begin{array}{lllll} \boxed{\text{product}} & A \times B & \rightsquigarrow & A \wedge B & \boxed{\text{conjunction}} \\ \boxed{\text{function}} & A \to B & \rightsquigarrow & A \Rightarrow B & \boxed{\text{implication}}. \end{array} \tag{7}$$

One basic characteristic of (classical) logic is that the conjunction $\wedge$ is **commutative**:

$$\mathsf{P} \wedge \mathsf{Q} \quad \Leftrightarrow \quad \mathsf{Q} \wedge \mathsf{P}. \tag{8}$$

This remains true of probabilistic logic, where $\wedge$ and $\vee$ are unified as conditional probability tables (CPTs) of the nodes of Bayesian networks. (Note: the commutative structure of $\wedge$ also gives rise to **monoidal categories**, that capture processes that can be executed in parallel; See [14] for an introduction.)

Once we know the symmetry, the question is how to impose this symmetry on deep neural networks. Interestingly, the answer already comes from an independent line of research (namely, PointNet [31] and Deep Sets [48]) that deals with visual object recognition of point clouds, eg. Fig. 1B.

In a point cloud, it does not matter the order in which the points are presented, as inputs to the classifier function. Such a function needs to be permutation invariant to a huge number of points. More generally, see also these recent articles on the use of geometry and symmetry in deep learning: [5] [6].

From [48]: the **Kolmogorov – Arnold representation theorem** states that every multivariate continuous function can be represented as a sum of continuous functions of one variable:

$$f(x_1, ..., x_n) = \sum_{q=0}^{2n} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right) \tag{9}$$

Their paper specialized the theorem to the case that every symmetric multivariate function can be represented as a sum of (the same) functions of one variable:

$$f(x_1, ..., x_n) = \rho(\ \phi(x_1) + ... + \phi(x_n)\ ) \tag{10}$$

This leads to the implementation using neural networks as in Fig. 3A, and can be easily implemented with just a few lines of Tensorflow, see §5.

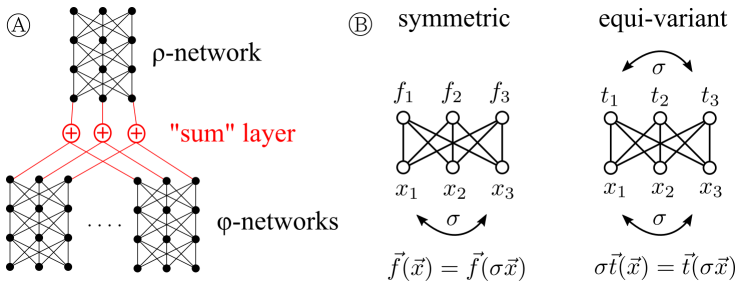


Fig. 3. Ⓐ symmetric neural network. Ⓑ permutation invariant vs. equivariant

The idea of using symmetric neural networks to process logical / relational data has already been explored by one of Google's research teams in 2017, which they called RN (Relational Networks) [34] [3]. Their results further confirm the viability of this idea.

3.1 Why BERT is a Logic

BERT (and its variants) are based on the Transformer architecture [8], and Transformers are based solely on the Self-Attention mechanism [43]. In Fig. 4 one can verify that the Transformer is **equivariant** to its inputs. That is to say, for example, if input #1 and #2 are swapped, then output #1 and #2 would also be swapped.

In other words, each Transformer layer takes N inputs and produces N equivariant outputs. That is the same as saying that *each* output is permutation-invariant in all its inputs. As we explained in the last section, permutation invariance is the symmetry that characterizes a logic as having *individual* propositions.

Proof that equi-variance $\Leftrightarrow$ symmetric (for an N-input N-output set function): $\Leftarrow$: Suppose we have constructed n symmetric functions $f_1, ..., f_N$, satisfying $\forall\sigma.\ \vec{f}(\vec{x}) = \vec{f}(\sigma\vec{x})$, with σ taking values in the symmetric group $\mathfrak{S}_N$. We can re-state the condition as $\forall\sigma.\ \sigma\vec{f}(\vec{x}) = \vec{f}(\sigma\vec{x})$ by re-naming the functions, because $\{f_i\}$ is a set. $\Rightarrow$: If we have N equi-variant functions $t_1, ..., t_N$, satisfying

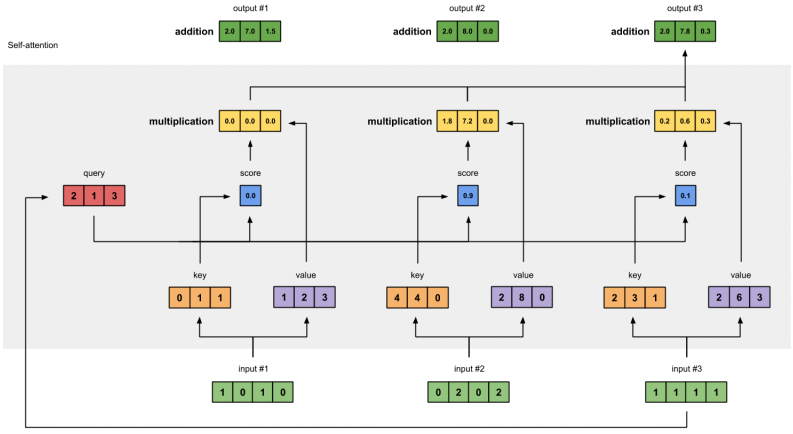


Fig. 4. Flow of operations in Self-Attention. From blog article: Illustrated: Self-Attention – Step-by-step guide to self-attention with illustrations and code https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

$\forall\sigma.\ \sigma\vec{t}(\vec{x}) = \vec{t}(\sigma\vec{x})$, we can also re-state the condition as $\forall\sigma.\ \sigma^{-1}\sigma\ \vec{t}(\vec{x}) = \vec{t}(\sigma\vec{x})$ by re-naming elements in the set $\{t_i\}$. This is illustrated in Fig. 3B (for $N = 3$). ■

A Self-Attention layer can be implemented by this formula (which appears in numerous tutorials):

$$Z = \sum \text{softmax}(\frac{1}{\sqrt{d_k}} Q_i \cdot K_j^T) V_j \tag{11}$$

where **Q**uery, **K**ey, and **V**alue are *linear* transformations of the inputs X_i by multiplying (learned) matrices W^Q, W^K, W^V respectively. We hypothesize that (11) is just a *general* form of equivariant functions, ie, it represents an arbitrary non-linear transformation of the input vectors X_i to the output vector Z, without any constraints other than equivariance.

Fig. 5C is a simplified view of a single Self-Attention layer. The output is a new proposition that depends on the input objects, and thus, functions as a **predicate**. However, this representation of predicates-within-proposition is not efficient for logic inference. We may visualize the non-linear (due to the softmax in Z) deformation of the input and output vector-embedding spaces as in Fig. 5D.

The problem is that a **universally quantified** formula such as $\forall x.P(x) \Rightarrow Q(x)$ requires mapping a source region to a target region in embedding spaces. This

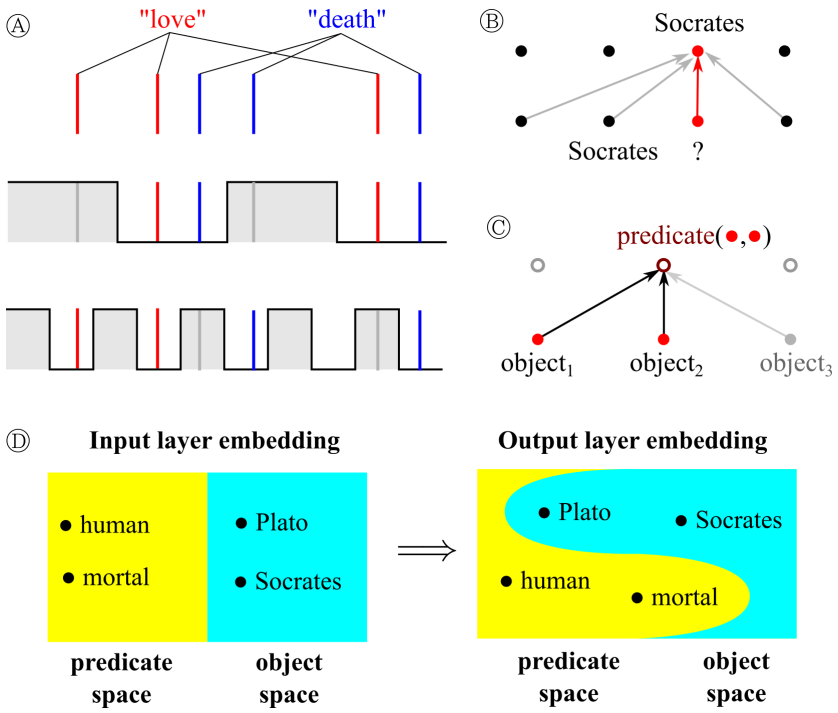


Fig. 5. Ⓐ Why Positional Encoding does not interfere with Word Embeddings. Ⓑ How a logic term "Socrates" is copied from one position to another. Ⓒ How predicates may be formed in Self-Attention. Ⓓ Non-linear deformation of embedding spaces.

kind of mapping shapes are difficult or slow to learn because it requires many pairs of input-output data points. But BERT/GPT is famous for being able to make **few-shot generalizations**. Thus we conjecture that in BERT/GPT the logical proposition is not just one equivariant unit but is **decomposed** into several units (eg. "I love you" at the input stage is decomposed into 3 vector units: "I", "love", and "you"). In other words, BERT/GPT performs logic inference / derivations on the **syntactic** level, ie, via **symbolic manipulations**.

Fig. 5B is an example of such an operation. The object "Socrates" is copied from one position to another. By looking at equation (11) we surmise that BERT is capable of such manoeuvres, with appropriately learned Keys and Values.

Fig. 5A tries to explain why Positional Encodings (the sine wave patterns) seem not to interfere with Word Embeddings, when the embedding dimension is sufficiently large. Note that the x-axis here is not a single dimension but many dimensions, and the sine waves are not ordinary waves but waves *over dimensions*. As each wave only occludes 50% of the dimensions, the word embeddings of "love" and "death" are still recognizable. This enables to mix multiple mean-

ings in a single word vector, eg. "love" + "you" = "love you". So a predicate vector may contain its own objects, as in Fig. 5C.

So far it seems the representation in BERT/GPT may have predicates with objects inside a single vector or with predicates and objects residing in separate vectors. Perhaps inspecting the weights inside BERT/GPT may reveal their internal representations.

In **Multi-Head Attention**, the intermediate computations are duplicated multiple (eg, $M = 8$) times, each with their own weight matrices. From the logic point of view, this amounts to duplicating M logic rules per output. But since the next layer still expects N inputs, the M outputs are combined into one, before the next stage. Thus, from the logic point of view this merely increased the parameters *within* a single logic rule, and seems not significant to increase the power of the logic rule-base. Indeed, experimental results seem to confirm that multi-head attention is not particularly gainful towards performance.

A comment is in order here, about the choice of the word "head". In logic programming (eg Prolog), one calls the conclusion of a logic rule its "head", such as `P` in `P :- Q,R,S`. Perhaps the creators of BERT might have logic rules in mind?

4 "No Free Lunch" Theory

In machine learning, "No Free Lunch" [45] [1] refers to the fact that accelerating the search for a solution by ignoring one part of the search space (known as "inductive bias" [1]) is just as good as ignoring another part, if the solutions are believed to be evenly distributed in those regions. For example, the symmetry proposed here reduces the search space by a factor of $1/n!$ where n is the number of propositions in working memory.

The following conceptual diagram of the algorithmic search space illustrates the possibility that there might exist some form of logic that is drastically different from the symbolic logic currently known to humans (Fig. 6).

but there is no efficient algorithm to find them (grey area is much larger than shaded area). The permutation symmetry proposed in this paper forces our logic to be decomposable into **propositions**. Such a logical form allows a mental state to be enumerated as a list of sentences (propositions), same as the "linear" structure of human **languages**. If the AGI knowledge representation is linear (in the sequential sense) and symbolic, then it would not be far from our formulation – all these logics belong to one big family.

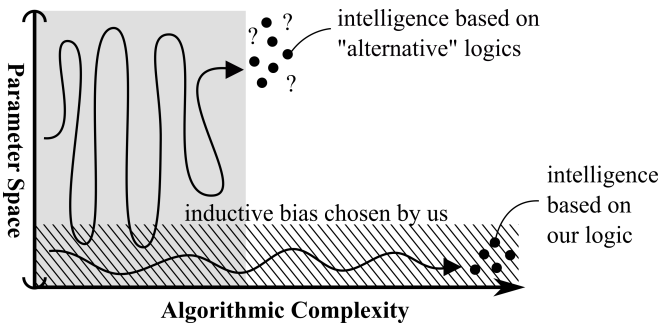


Fig. 6. Inductive bias and the search for AGI.

But could there be drastically different logics? One observes that pictures and music are not easily described by words, indeed they are 2-dimensional structures. This suggests that the brain may use **multi-dimensional** arrays of features to represent the world. Such a "logic" would be very different from sequential logic and it would be interesting and fruitful to analyze the relation between them.

5 Experiment

A simple test [6] of the symmetric neural network, under reinforcement learning (Policy Gradient [7]), has been applied to the Tic-Tac-Toe game.

The state of the game is represented as a set of 9 propositions, where all propositions are initialized as "null" in the beginning. During each step of the game, a new proposition is added to the set (ie. over-writing the null propositions). Each proposition encodes who the player is, and which square (i, j) she has chosen. In other words, it is a predicate of the form: `move(player,i,j)`. The neural network takes 9 propositions as input, and outputs a new proposition; Thus it is a permutation-invariant function.

In comparison, the game state of traditional RL algorithms (eg. AlphaGo [37] [38] [30]) usually is represented as a "chessboard" vector (eg. 3×3 in Tic-Tac-

[6] Code with documentation and more detailed analysis is on GitHub: https://github.com/Cybernetic1/policy-gradient

[7] The Policy Gradient algorithm is chosen because it allows *continuous* actions. Other reinforcement learning algorithms require learning the value function over actions, and when the action space is not discrete such a value function cannot be represented by a table, but perhaps as a neural network. However, it is not easy to find the *maximum* of a neural network, which is required to choose the optimal action. Policy Gradient avoids this because the policy function directly maps to actions.

Toe, 8×8 in Chess, $19 \times 19 = 361$ in Go [8]). This state vector is the same constant length even if there are very few pieces on the chessboard. Our logic-based representation may offer some advantages over the board-vector representation, and likely induces a different "way of reasoning" about the game.

In our Tic-Tac-Toe experiment, learning led to initial improvements in game play but failed to achieve the optimal score in general. We find that this failure is also shared by the fully-connected NN (neural network), and this is likely because the policy gradient algorithm itself does not converge for Tic Tac Toe. Fig. 7 is a comparison of symmetric NN versus fully-connected NN during early training. Disappointingly, the symmetric version does not out-perform the fully-connected version.

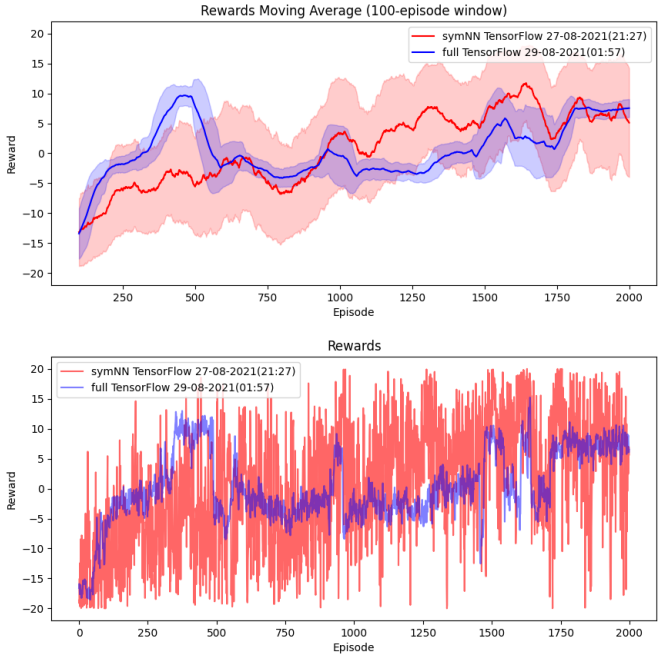


Fig. 7. Symmetric vs. fully-connected neural network for Tic Tac Toe

We ascribe this failure to the naive policy gradient algorithm and plan to use Actor-Critic (which also allows continuous actions) in our next experiments. We hope to show that symmetric NN is gainful for solving problems with logical

[8] In AlphaGo and AlphaZero, the algorithm makes use of several auxiliary "feature planes" that are also chessboard vectors, to indicate which stones have "liberty", "ko", etc.

structure. In another Github experiment we explore using a symbolic logic engine to solve Tic Tac Toe [9] and the comparison of these two approaches may shed light on how to integrate deep learning with logic.

6 Conclusion and Future Directions

We described a minimal AGI with a logic that can derive one new proposition per iteration. This seems sufficient to solve simple logic problems such as Tic-Tac-Toe. As a next step, we would consider inference rules with multi-proposition conclusions. The latter seems essential to **abductive** reasoning. For example, one can deduce the concept "apple" from an array of visual features; Conversely, the idea of an "apple" could also evoke in the mind a multitude of features, such as color, texture, taste, and the facts such as that it is edible, is a fruit, and that Alan Turing died from eating a poisoned apple (a form of episodic memory recall), and so on. This many-to-many inference bears some similarity to the brain's computational mechanisms [32] [33] [4]. The author is embarking on an abstract unifying AGI theory that makes references to (but not necessarily copying) brain mechanisms.

Acknowledgements

Thanks Ben Goertzel for suggesting that neural networks are advantageous over pure symbolic logic because they have fast learning algorithms (by gradient descent). That was at a time when "deep learning" was not yet a popular word. Thanks Dmitri Tkatch for pointing me to existing research of symmetric neural networks. Thanks Dr. 肖达 (Da Xiao) for explaining to me details of BERT.

Also thanks to the following people for invaluable discussions over many years: Ben Goertzel, Pei Wang (王培), Abram Demski, Russell Wallace, Juan Carlos Kuri Pinto, SeH, Jonathan Yan, and others. Also thanks to all the university professors and researchers in Hong Kong (especially in the math departments, and their guests), strangers who taught me things on Zhihu.com (知乎), Quora.com, and StackOverflow.

References

1. Alpaydin, E.: Introduction to machine learning. MIT press (2020)

[9] https://github.com/Cybernetic1/GIRL

2. Baez, J., Stay, M.: Physics, topology, logic and computation: a rosetta stone. In: New structures for physics, pp. 95–172. Springer (2010)
3. Battaglia, et al.: Relational inductive bias, deep learning, and graph networks https://arxiv.org/pdf/1806.01261.pdf
4. Boraud, T.: How the brain makes decisions. Oxford (2020)
5. Bronstein, M.: Geometric foundations of deep learning (2021), https://towardsdatascience.com/geometric-foundations-of-deep-learning-94cdd45b451d
6. Bronstein, M.M., Bruna, J., Cohen, T., Velickovic, P.: Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. CoRR **abs/2104.13478** (2021), https://arxiv.org/abs/2104.13478
7. Bělohlávek, Dauben, Klir: Fuzzy logic and mathematics: a historical perspective. Oxford University Press (2017)
8. Devlin, Chang, Lee, Toutanova: Bert: pre-training of deep bidirectional transformers for language understanding (2018), arXiv:1810.04805v2[cs.CL]
9. Evans, Grefenstette: Learning explanatory rules from noisy data. Journal of artificial intelligence research (2017)
10. Feng, K., Qin, M.: Symplectic geometric algorithms for Hamiltonian systems. Springer (2010)
11. Goertzel, B.: The general theory of general intelligence: A pragmatic patternist perspective. arXiv preprint arXiv:2103.15100 (2021)
12. Goertzel, B., Duong, D.: Opencog ns: A deeply-interactive hybrid neural-symbolic cognitive architecture designed for global/local memory synergy. In: 2009 AAAI Fall Symposium Series (2009)
13. Harris, J., Morrison, I.: Moduli of curves, vol. 187. Springer Science & Business Media (2006)
14. Heunen, C., Vicary, J.: Categories for Quantum Theory: an introduction. Oxford University Press (2019)
15. Hinton, G.: How to represent part-whole hierarchies in a neural network. arXiv preprint arXiv:2102.12627 (2021)
16. Hitzler, Seda: Mathematical aspects of logic programming semantics. CRC Press (2011)
17. Höhle, U.: Fuzzy sets and sheaves. part i: basic concepts. Fuzzy Sets and Systems **158**(11), 1143–1174 (2007)
18. Höhle, U.: Fuzzy sets and sheaves. part ii:: sheaf-theoretic foundations of fuzzy set theory with applications to algebra and topology. Fuzzy Sets and Systems **158**(11), 1175–1212 (2007)
19. Hutter, M.: Universal artificial intelligence. Springer (2005)
20. Jacobs, B.: Categorical logic and type theory. Elsevier (1999)
21. Jardine: Fuzzy sets and presheaves (2019), arXiv:1904.10314v5[math.CT]
22. Kappen: An introduction to stochastic control theory, path integrals and reinforcement learning. AIP conference proceedings 887 (149) (2007), https://doi.org/10.1063/1.2709596
23. Leimkuhler, B., Reich, S.: Simulating hamiltonian dynamics. No. 14, Cambridge university press (2004)
24. Liberzon, D.: Calculus of variations and optimal control theory: a concise introduction. Princeton Univ Press (2012)
25. MacLane, S., Moerdijk, I.: Sheaves in geometry and logic – a first introduction to topos theory. Springer (1992)
26. Mann, P.: Lagrangian and Hamiltonian dynamics. Oxford University Press (2018)

27. Nordström, Petersson, Smith: Martin-Löf's Type Theory, vol. 5. Oxford University Press (2000)
28. Potapov, A., Belikov, A., Bogdanov, V., Scherbatiy, A.: Cognitive module networks for grounded reasoning. In: International Conference on Artificial General Intelligence. pp. 148–158. Springer (2019)
29. Program, T.U.F.: Homotopy type theory: Univalent foundations of mathematics (2013)
30. Pumperla, Ferguson: Deep learning and the game of Go. Manning (2019)
31. Qi, Su, Mo, Guibas: Pointnet: Deep learning on point sets for 3d classification and segmentation. CVPR (2017), https://arxiv.org/abs/1612.00593
32. Rolls, E.: Cerebral cortex – principles of operation. Oxford (2016)
33. Rolls, E.: Brain computation – what and how. Oxford (2021)
34. Santoro, A., Raposo, D., Barrett, D.G.T., Malinowski, M., Pascanu, R., Battaglia, P., Lillicrap, T.: A simple neural network module for relational reasoning (2017)
35. Schlichenmaier, M.: An introduction to Riemann surfaces, algebraic curves and moduli spaces. Springer Science & Business Media (2010)
36. Shalev-Shwartz, S., Ben-David, S.: Understanding machine learning: From theory to algorithms. Cambridge university press (2014)
37. Silver, e.a.: Mastering the game of go with deep neural networks and tree search. Nature pp. 484–489 (2016)
38. Silver, e.a.: Mastering the game of go without human knowledge. Nature pp. 354–359 (2017)
39. Simmons: Derivation and computation: taking the Curry-Howard correspondence seriously. Cambridge University Press (2000)
40. Sørensen, Urzyczyn: Lectures on the Curry-Howard isomorphism. Elsevier (2006)
41. Sutton: Temperal credit assignment in reinforcement learning (1984)
42. Thompson: Type theory and functional programming. Addison-Wesley (1991)
43. Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, Polosukhin: Attention is all you need (2017), https://arxiv.org/abs/1706.03762
44. Vickers: Fuzzy sets and geometric logic. Fuzzy sets and systems (161), 1175–1204 (2010)
45. Wolpert, Macready: No free lunch theorems for optimization. IEEE transactions on evolutionary computation 1 (67) (1997)
46. Yan: AGI logic tutorial (2021), https://drive.google.com/file/d/1v2efrH4gVJS9wG-KKgi1uFbbCoMOc9Hl/view?usp=sharing
47. Yan, K.Y.: Fuzzy-probabilistic logic for common sense reasoning. Artificial general intelligence 5th international conference, LNCS 7716 (2012)
48. Zaheer, Kottur, Ravanbakhsh, Schneider, Poczos, Salakhutdinov, Smola: Deep sets. NIPS 2017 (2017), https://arxiv.org/abs/1611.04500