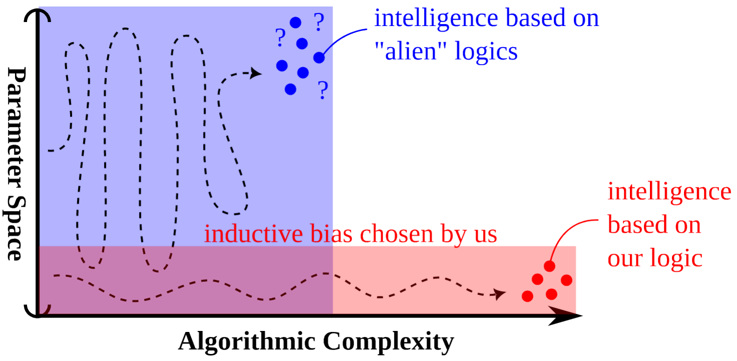
Logic for AGI Speed-Up

YKY [November 9, 2025]

My research in AGI has been mainly focused on using the structure of logic to constrain the **hypothesis space** of AGIs along the line of **No Free Lunch** and **inductive biases**. The idea is that if we know all AGIs to share a certain logical structure, then we can use this structure to limit the search space (ie. hypothesis space), thus learning will be accelerated. And learning (ie. AGI training) is very expensive. This can be illustrated by the following diagram:


intelligence based on "alien" logics
?
?
?
Parameter Space
inductive bias chosen by us
intelligence based on our logic
Algorithmic Complexity

(1)

Richard Sutton (I don't know him personally, but I think I am one of his most loyal disciples in reinforcement learning) seems to disagree with me on this issue. He thinks we are much more likely to find intelligences outside of our notions of logic. Can the blue area be smaller than the red area? I don't have a strong reason to refute him, but my intuition tells me to pursue my idea further...

Usually "**structure**" in mathematics is expressed as some kind of **symmetry**. Usually symmetry is expressed as some kind of equation. For our purpose, for logic, the most relevant symmetries are:

$$I\heartsuit U \neq U\heartsuit I \quad \text{(else there wouldn't be heartbreaks)}$$
$$I\heartsuit U \wedge U\heartsuit I = U\heartsuit I \wedge I\heartsuit U \tag{2}$$

The second equation describes the **commutativity** of conjunctions of propositions, one of the most celebrated symmetries in mathematics (The adjective **Abelian** means commutative, such as $ab = ba$ in group theory, named after the Norwegian mathematician Abel). This symmetry is also possessed by the Transformer / **Self-Attention** (that's why we need to add **positional encoding** to the inputs to Transformers).

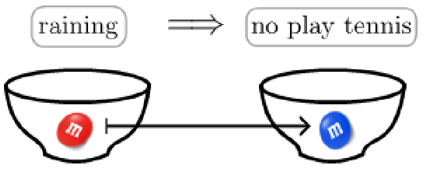
If we look more closely at Self-Attention, we realize it has some sort of *mismatch* with logical structure. As words correspond to tokens, and sentences correspond to logic propositions, it should be the sentences that are commutative, not the words. Is this mismatch very significant for machine learning, or is it computationally trivial? I still don't know as of now. For a long time I'd wanted to fix this mismatch but it turned out to be very difficult on a theoretical level.

If we could encapsulate this structure (non-commutative at the predicate level, commutative at the propositional level) into an **algebraic** structure, such as × being non-commutative and + being commutative, then we could perform our machine learning within that algebra. And thanks to the **representation theory** of algebras, this could be transformed into operations with matrices. But this path turns out to be unfeasible as I somehow failed to "fit" logic into an algebra. In fact, the **algebraization** of logic is a very complicated research topic that has occupied great logicians such as Alfred Tarski and Paul Halmos, among others, resulting in formulations such as cylindrical algebra, relation algebra, etc. We all know from high school, that Boolean logic with ∧ and ∨ behaves like an algebra, and can indeed be turned into a **Boolean ring**, where ×

is AND but + is exclusive-OR. So far so good, but as soon as we move to **predicate logic** we seem to run out of naturally "algebraic" ideas to represent predicates. Halmos suggests to treat predicates as **algebra homomorphisms**, "homo" meaning the morphisms preserve algebraic structure. In my (unfinished) master's thesis I was half-way working on this approach.

Another path is through **categorical logic**, which I have spent >15 years learning. I know that logic can be captured by a category, more specifically a **topos**. But I didn't realize that *this* is the very representation that I've been looking for, day after day for 15+ years, until I had a chat with GPT!

To understand categorical logic, we must start with the **Curry-Howard correspondence**. It basically says, a **logical implication** such as $A \Rightarrow B$ should be interpreted as a **function** $f : A \to B$, ie. from the space A to the space B, mapping a **proof object** in A to another proof object in B. Think of a proposition as a bowl, each may or may not have a piece of M&M in it:



(3)

This is rather peculiar, as propositions are interpreted as "spaces," or even more abstractly as **types**, as in type theory, which some programmers or computer science majors may be familiar with.

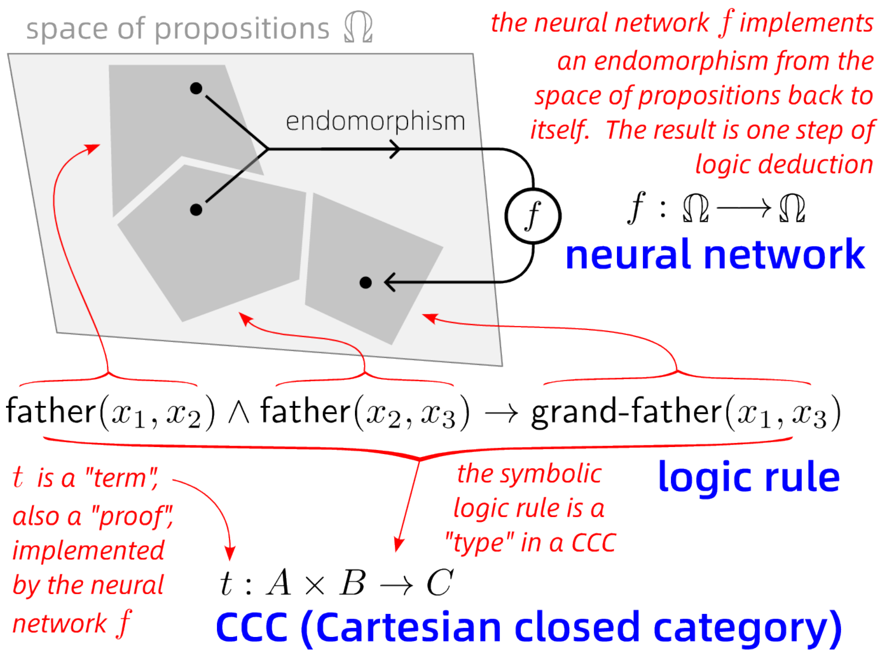
The Curry-Howard correspondence is not something that can be proven; it is more like an axiom. In the **philosophy** of mathematics, one studies why people have notions of numbers like 1,2,3... which to me are rather boring questions (I much prefer problems like $\mathrm{P} \overset{?}{=} \mathrm{NP}$), but that's where Curry-Howard belongs. It is one of the *profoundest* mathematical discoveries, that links logic to mathematics on a meta-physical level.

It has been re-discovered so many times that its full name could be Brouwer-Heyting-Kolmogorov-Schönfinkel -Curry-Meredith-Kleene-Feys-Gödel-Läuchli-Kreisel-Tait-Lawvere-Howard-de Bruijn-Scott-Martin-Löf-Girard-Reynolds-Stenlund-Constable-Coquand-Huet-Lambek...

(A few of them are still breathing as of 2025. The programming language Haskell is named after Haskell Curry (#5)).

Lambek (#24) discovered the link between **higher-order logic** (HOL) and **Cartesian-Closed Categories** (CCCs), which I find very useful. This is a form of "untyped" logic which suffers ffrom Curry's paradox (#5) but which I think can be circumvented by fuzzy logic. Bertrand Russell and Alfred North Whitehead invented **type theory** to get around Curry's paradox, but I personally prefer untyped logic.

To *apply* Curry-Howard to AGI, perhaps the most important insight is explained by the following figure, which links neural networks to logic and to CCCs:



space of propositions Ω
endomorphism
the neural network f implements an endomorphism from the space of propositions back to itself. The result is one step of logic deduction
$f : \Omega \longrightarrow \Omega$
neural network
$\mathsf{father}(x_1, x_2) \wedge \mathsf{father}(x_2, x_3) \to \mathsf{grand\text{-}father}(x_1, x_3)$
logic rule
t is a "term", also a "proof", implemented by the neural network f
the symbolic logic rule is a "type" in a CCC
$t : A \times B \to C$
CCC (Cartesian closed category)

(4)

The neural network, which is a non-linear **function**, is seen as a **term** belonging to a **type** which is the logic formula. This accords beautifully with Curry-Howard. A proposition such as father(a,b) is a space, and all such spaces form the big space Ω which is not itself a proposition; that's why I use a different font for it.

A proposition such as ♡(Romeo, Juliet) is created by passing the ordered pair (Romeo, Juliet) to the predicate ♡, which outputs a proposition, ie. a space, in Ω. In other words, $\heartsuit : A \times A \to \Omega$, where A is the set of first-order objects, eg. the set of people. ♡ is called a **type-constructor**; it creates new types (propositions) out of existing types (propositions). The keen reader may notice that the arrow → is *overloaded* with two purposes: one as logic implication ⇒ as required by Curry-Howard, the other as type constructor. We can read the arrow in ♡ as "if you show me a pair of who is a boy and who is a girl, I can show you if he loves her" – a very awkward sentence, but a sentence nonetheless. This "rescues" the *consistency* of the Curry-Howard correspondence, and is known as **Martin-Löf type theory** (#17).

Does CCC really capture the two-layer non-commutative and commutative structure in (2)? In category theory the products $A \times B$ and $B \times A$ are *isomorphic* but *not equal*. If we interpret logic ∧ (AND) as ×, it seems to be non-commutative. But if we look at whether there exists a *function* that takes one object in each of A, B and maps them to the target, then the order of A and B does *not* matter, and quite miraculously I think, we got the commutativity of logic ∧.

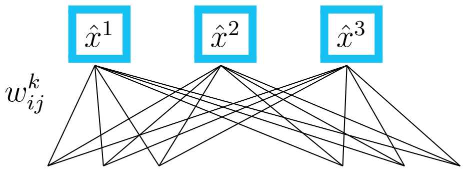
On the other hand, for the constructed type ♡(Romeo, Juliet), we can let the type constructor ♡ simply create the Cartesian product $V_\heartsuit \times V_R \times V_J$ where each V is a vector space. This is similar to the concatenation of **tokens** in Transformers, except we are concatenating vector spaces instead of vectors. And $V_\heartsuit \times V_R \times V_J \neq V_\heartsuit \times V_J \times V_R$. Thus we have non-commutativity at the predicate level.

At this point we are ready to define the **type** where our neural network f lives in, under the Curry-Howard framework:

$$f : \overbrace{(\Omega \to 2)}^{\text{set of propositions}} \to \overbrace{(\Omega \to \mathbb{R})}^{\text{probabilities over propositions}}. \tag{5}$$

The left side contains a set of propositions and the right side contains just one output proposition (but it must have probabilities distributed over it). Both parts are so huge that they must be decomposed using appropriate tricks. This is where our neural network shares similarities with **Transformers**: the left side requires **permutation invariance** such as offered by Self-Attention, the right side requires sentences to be decomposed into "**tokens**," with probabilities distributed over them.

Another extremely complicated issue we have not touched upon, is that of **variables** and their **substitutions**. Alonzo Church invented λ-calculus in the 1930s in order to study the mechanisms of variable substitution explicitly. Long story short, I formulated a way to do variable substitutions using neural networks, in my unfinished master thesis. I was following Paul Halmos' algebraic logic approach (which does not have Curry-Howard in it), but my result can also fit into the Curry-Howard framework. The neural network looks like this:

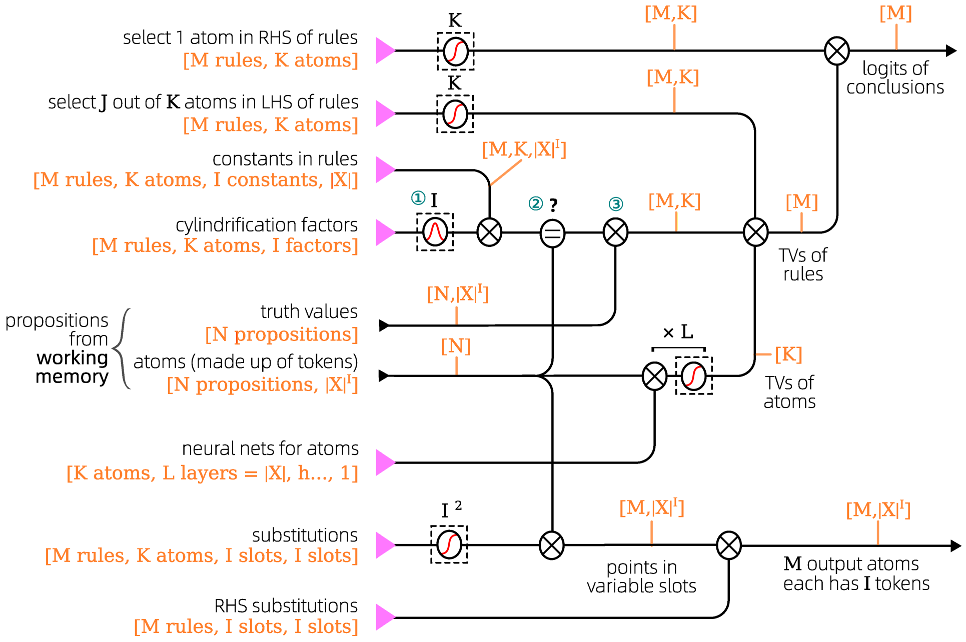


$$\mathsf{father}(x_{11}, x_{12}, \Box_{13}) \wedge \mathsf{father}(\Box_{21}, x_{22}, x_{23}) \rightarrow \mathsf{grand\text{-}father}(x_1, \Box_2, x_3) \tag{6}$$

The variable slots are on top, and they are shared by multiple propositions below. Here we need **Softmax** to *select a dominant object* to occupy a slot. This mechanism is very similar to Self-Attention.

再剩下的问题是：

- 神经网络的空间大小、其可不可以叠加的问题
- rules matching 问题，则又回到 my thesis 卡在的点上
-



(7)

And yes, I am aware of the exploitation of cocoa farmers in Africa.