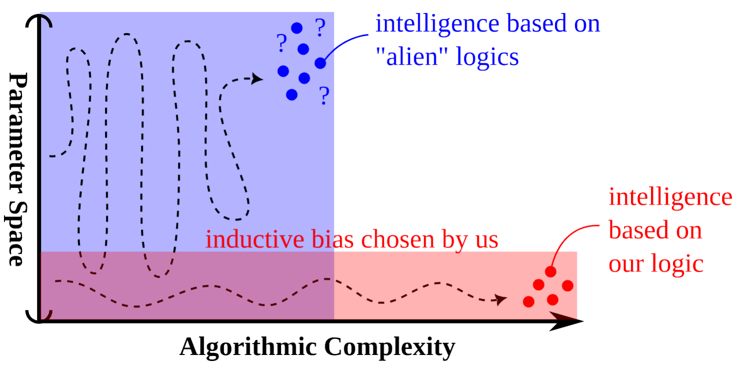
Logic for AGI Speed-Up

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:

 (1)
intelligence based on "alien" logics
Parameter Space
inductive bias chosen by us
intelligence based on our logic
Algorithmic Complexity

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).

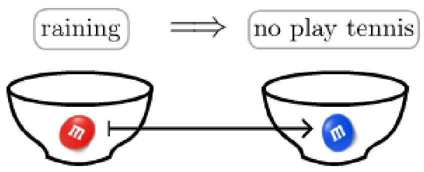
If we look more closely at Self-Attention, we realize it has some sort of *mismatch* with logical structure. As words correspond to tokens, and sentences correspond to logic propositions, it should be the sentences that are commutative, not the words. Is this mismatch very significant for machine learning, 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 $\times$ 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 $\wedge$ and $\vee$ behaves like an algebra, and can indeed be turned into a **Boolean ring**, where $\times$

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.

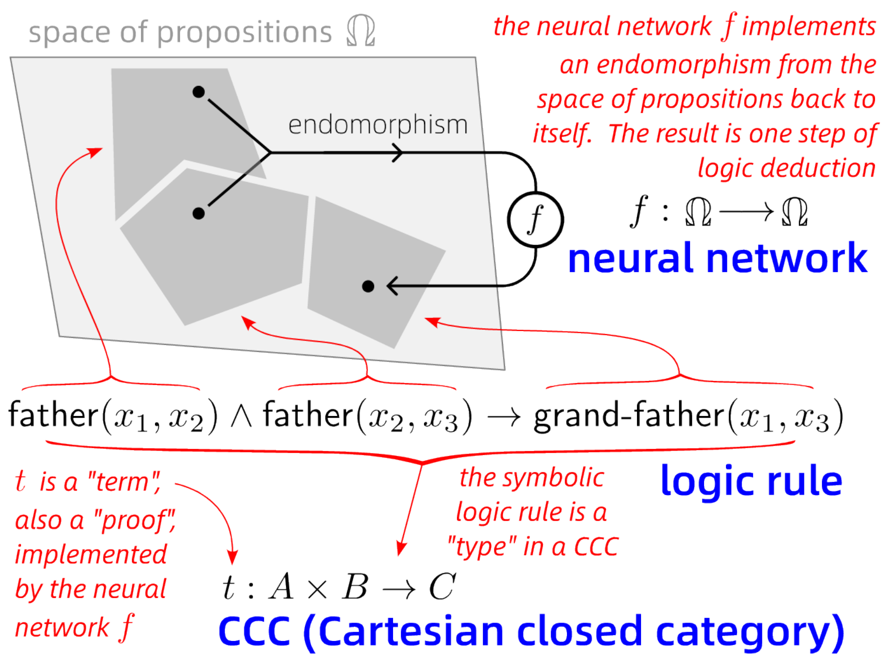
The Curry-Howard correspondence is not something that can be proven; it is more like an axiom. In the **philosophy** of mathematics, one studies why people have notions of numbers like 1,2,3... which to me are rather boring questions (I much prefer problems like P $\stackrel{?}{=}$ NP), but that's where 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 guy #5.)

Lambek (guy #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 (guy #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
f
the neural network f implements an endomorphism from the space of propositions back to itself. The result is one step of logic deduction
f : Ω ⟶ Ω
neural network
father(x₁, x₂) ∧ father(x₂, x₃) → grand-father(x₁, x₃)
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 × B → 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 to denote 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 $\to$ is *overloaded* with two purposes: one as logic implication $\Rightarrow$ 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** (guy #17).

Does it really work? Does it 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 $\wedge$ (AND) as $\times$, 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 $\wedge$.

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.

进一步确定 神经网络 f 的形式。我刚说 CCC 形式 固定了 props 的交换性，那么为什么又要再一次「做」交换性出来？

如果 交换性 和 substitutions 都已经搞掂，那还等什么呢？

以前那个网络就是 props-in props-out 的形式。记得好像 props 是以 subject space X 内的 vectors 形式 表示的。但现在 props 是某些空间。

它不是 vector，但可以看成是 vector 周围的空间？这在我 2024 发表的论文里已经确立了。NN 将 vector maps to vector 没有问题 – 那完全是正常、且符合 Curry-Howard 的。问题是现在 props 可以根据 vector embedding 构造出来，这是新的。所谓旧的方法是什么？GPT？GPT 的输出是 probability distro over words，其实符合 Curry-Howard?? 反而是要输出 props 可能 unfeasible。这也是之前已经有的结论了。

问题是 从 input props 来看，它们的交换性好像要额外处理，why？这可能是所谓 power set 的问题。如果只考虑一个 prop 的输入，它存在于 Ω 里面，其实即是 tok $\times$ tok $\times$ tok ... 的空间。Transformer 用了 Attention 去处理 tokens 的交换性。然则我们刚刚不是说好了 CCC 自带交换性？交换是指... 当我们有两个或以上的 props，这其实是怎样表示的？显然，它是 props $\rightarrow 2 = 2^{\text{props}}$ 的形式。换句话说，每个 prop 附带 $\{\top, \bot\}$ 的讯息。那也可以看成是一堆被选择了的 props，但这些 props 是否可交换的呢？如何决定？每组被选择的 props 构成一个 subset，它就是一个不可分割的 element。然后这个 subset element 被 map 到目标。但这是否可以在神经网络做到？

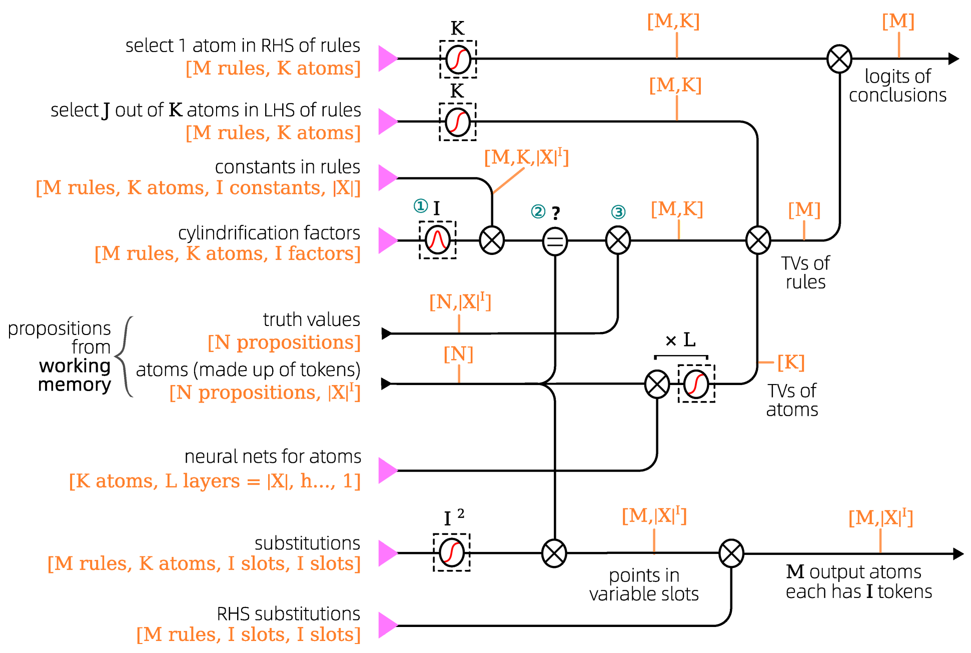
每一次 Self Attention，所有 tokens 齐齐送上，然后输出 memory table lookup 的叠加。这里面已经融合了 matching 和 apply，但它的结构里面已含有 交换性。

如果说，需要的 maps 是由集合到集合，则当然可以满足交换性，但似乎也不见得 范畴论特别有用。也就是说，即使固定了 type，仍然是由 terms 决定 函数的交换性。Type 里面仍然可以存在某些 terms 利用 ordered pairs 保持 非交换性。Curry-Howard 好像是一个空壳，它给予了 交换性 的可能，而且提供了一个结构的框架，也算是有点用吧。

现在从 Curry-Howard 的框架，再从新定义一次

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
-



(5)