AGI standard model
— trying to establish a consensus

YKY

June 16, 2022

Abstract

The essence of the standard model is just to identify a **Working Memory** as the "state" of the AGI system. This establishes connections between reinforcement learning, Turing machines, neural Turing machines, self-attention, Transformers, and even the biological brain. Lastly, we describe two subproblems: abductive reasoning and making assumptions, that also seem essential to AGI systems.

0 Introduction

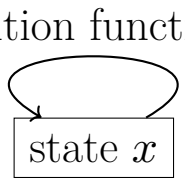
The "standard model" is a way of thinking, that may help us better understand the general theory of AGI systems.

The essence of the standard model is just to identify a **Working Memory** as the "state" of the AGI system.

One benefit of our theory is that it relates Transformers / BERT / GPT to AGI systems. These language models are phenomenally intelligent, yet many people criticize them as not "truly" intelligent. The standard model suggests that they are indeed linked to AGI.

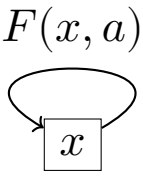
1 Reinforcement learning

This is the simplest form of a **dynamical system**:



$= $ "working memory" (1)

When we add a "control" or "action" variable a to it, it becomes the most basic **control system**:



(2)

which is the setting for Dynamic Programming or **Reinforcement Learning**. The optimal solution for such systems is governed by the **Hamilton-Jacobi-Bellman equation**:

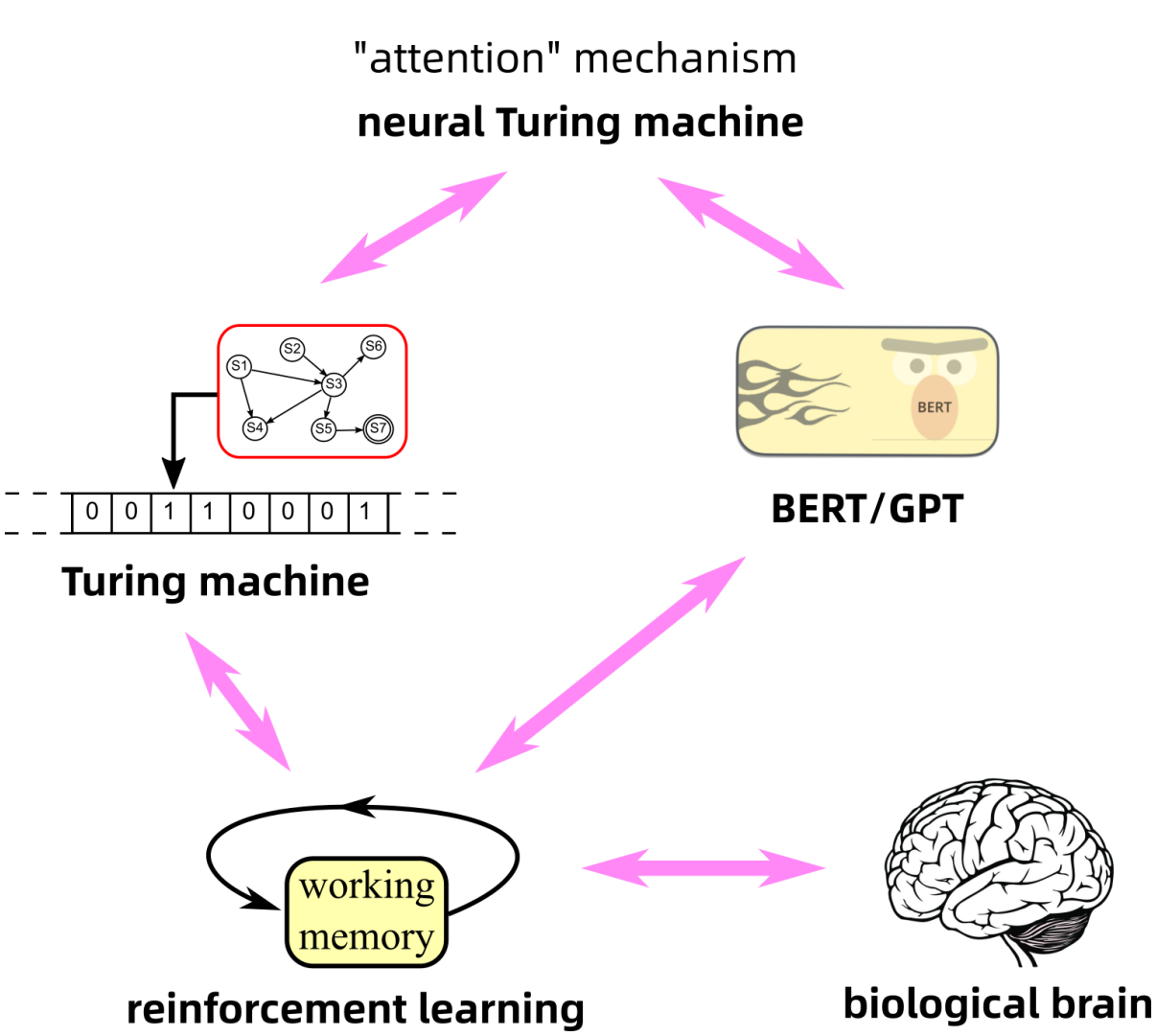
$$V_t^* = \max_a \mathbb{E}[R + V_{t+1}^*] \tag{3}$$

Here I want to ask a question: reinforcement learning is not

Recently, Yann LeCun's **Energy-Based Models** offers a way to circumvent the problem of learning probability distributions over actions, when the action space is hugely high-dimensional. This seems to be an important step towards AGI systems.

I call this the "standard model" because of the extreme simplicity of this setup, and that I don't know of other alternative models that deviate much from it.

The following diagram shows how the standard model relates to several other important areas, so we can reap profits from their interactions:

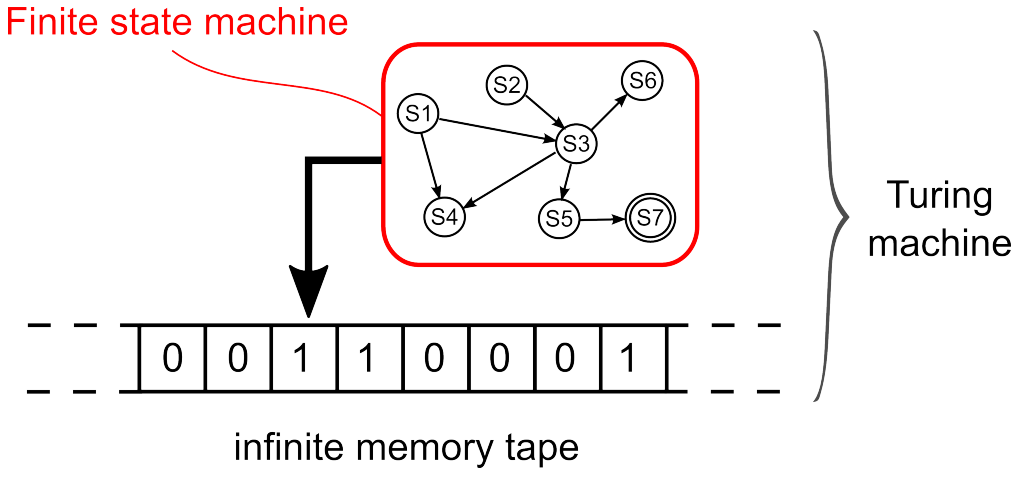


(4)

2 Neural Turing Machines and Transformers

The **attention mechanism** was first proposed in the "**Neural Turing Machine**" paper by Graves *et al* [2014].

Recall that a Turing machine is a **Finite State Machine** augmented with a **Memory Tape**:



(5)

In Neural Turing Machines, Graves *et al* proposed the attention mechanism for an RNN "Controller" (playing the role of the Finite State Machine) to read and write from a **Memory Matrix** (the tape), using a content-based addressing method.

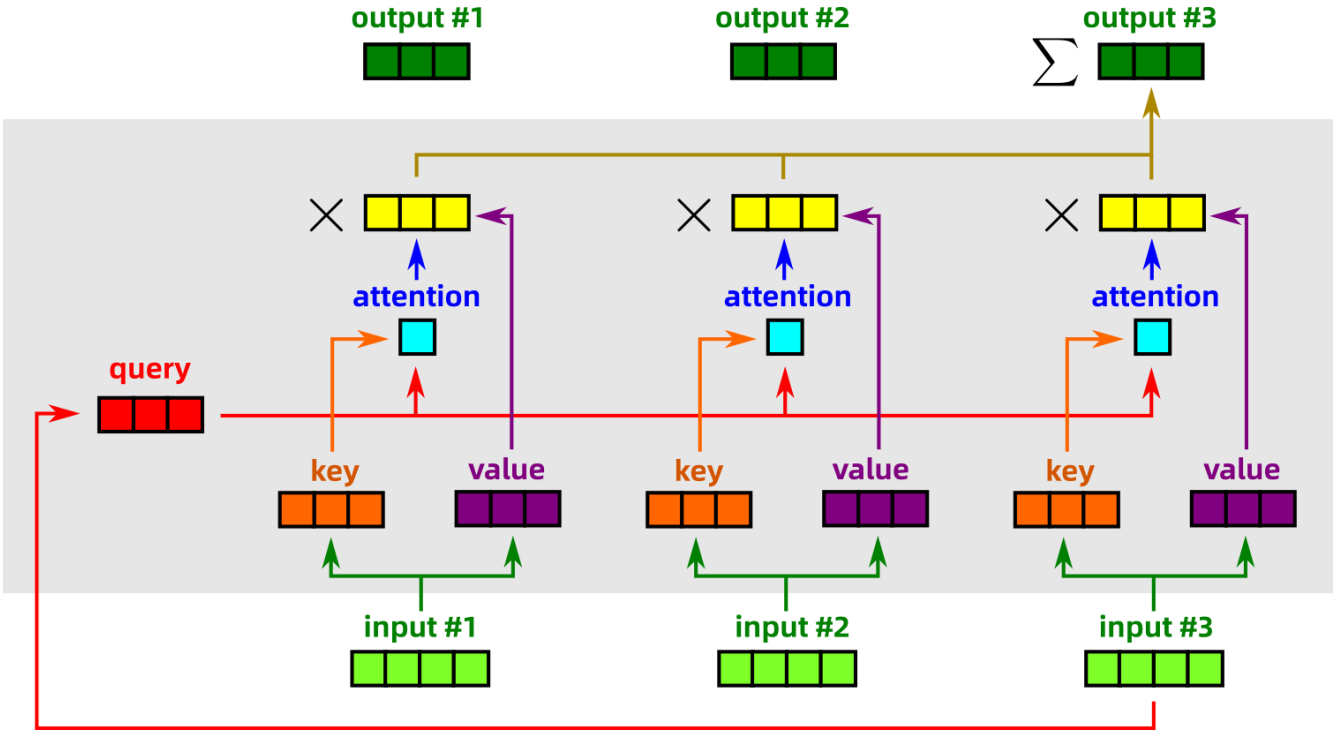
The Memory Matrix M consists of N items, each of constant size. *The discreteness of the address would introduce discontinuities in gradients of the output*, hence we need an **Attention Vector** to focus on a specific location in the memory matrix M.

The Attention Vector $\vec{a}$ is calculated via the following formula, familiar to students of the Transformer:

$$\vec{a} = \text{soft}\max_i\{\mathcal{D}(K, M_i)\} \tag{6}$$

where $D()$ is a similarity measure between the key K and memory item M_i. The key K is emitted by the Controller as the value that it is looking for.

This then evolved into the **Self Attention** mechanism used in all Transformers. Now let us refresh with this diagram illustrating Self-Attention (redrawn from a blog article on the web):



(7)

The research done by Olah *et al*, in their 2021 paper *A Mathematical Framework for Transformer Circuits*, is very helpful towards understanding Transformers

and Self-Attention. The paper is technically quite challenging, but thanks to the guidance of professor Xiao Da from Beijing I was able to understand the main ideas. Here I offer some pointers to help others understand the paper, without explaining it in full details.

The Self Attention A is calculated by matrix operations followed by **softmax**, and thus is non-linear. However, when A is held constant, the Transformer becomes a linear operation that can be expressed as **tensor products**, for example:

$$T = \underbrace{\mathrm{Id} \otimes W_U}_{\text{The token unembedding maps residual stream vectors to logits.}} \cdot \underbrace{\left(Id + \sum_{h \in H_1} A^h \otimes W_{OV}^h\right)}_{\text{The attention layer has multiple heads. The result of each is added into the residual stream.}} \cdot \underbrace{\mathrm{Id} \otimes W_E}_{\text{The token embedding maps tokens to residual stream vectors.}} \tag{8}$$

For example, to understand the equation for a Transformer "Head":

$$h(x) = (\mathrm{Id} \otimes W_O) \cdot (A \otimes \mathrm{Id}) \cdot (\mathrm{Id} \otimes W_V) \cdot x \tag{9}$$

it is helpful to know that x is a **matrix** with dimensions $[n_{\text{context}}, d_{\text{model}}]$, ie, one dimension being the number of tokens and the other dimension representing the token's vector. When a tensor such as $U \otimes V$ acts on x, U acts on the "position" dimension of the matrix while V acts on the "vector" dimension. So the above equation can be collapsed to:

$$h(x) = (A \otimes W_O W_V) \cdot x. \tag{10}$$

Thus tensor products are very convenient for expressing such operations.

Tensor product

Recall that a **linear** map is defined between two vector spaces:

$$F : U \to V. \tag{11}$$

Similarly one can define a **bi-linear** map:

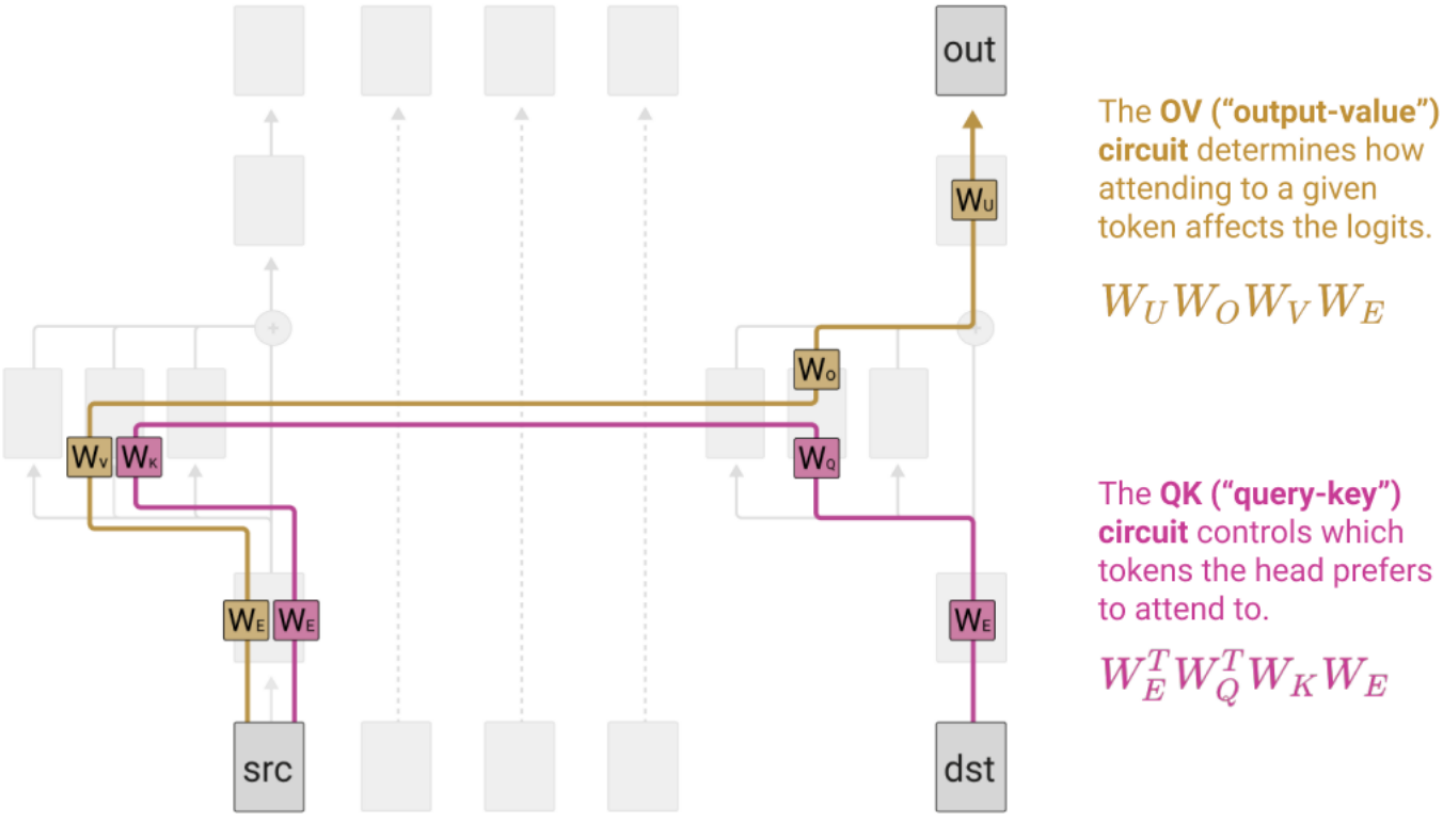
$$\Phi : V \times W \to U \tag{12}$$

that is linear in both its first and second arguments. The tensor product $\otimes$ is the **universal** multi-linear map.

The tensor product generalizes to the mathematically important **monoidal categories**. The latter captures the associativity (and optionally commutativity, for which we have symmetric monoidal categories), but the linearity is abstracted away.

Monoidal categories can be depicted by **String Diagrams**. There are also **Neural String Diagrams** that describe the structure of neural networks. For example, the Transformer layer is **equivariant** in its inputs.

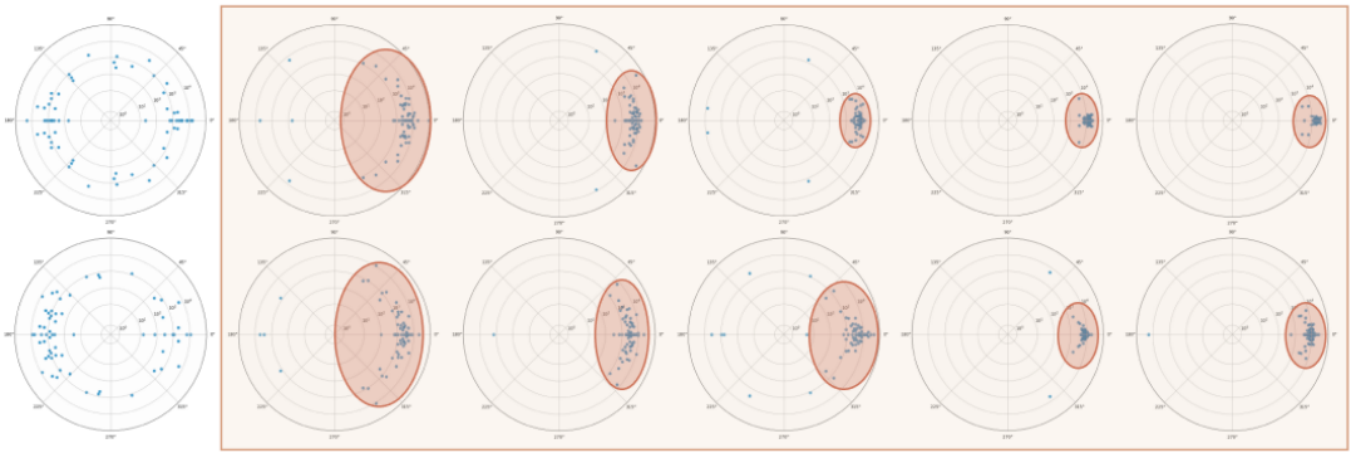
In the Transformer, we can see **Input Values** flowing through some "circuits" defined by matrices to yield the **Output Values**:



(13)

Some Attention Heads merely perform "**copying**" of a vector from a previous

layer to the next. If we spell this out as a linear equation, $W\vec{x} = \lambda\vec{x}$, where the transformation W merely results in a **scaling** of the vector $\vec{x}$. This is just an **eigenvector** equation. So if we plot the eigenvalues of the transformation matrices on the complex plane, the "copy" maps would have eigenvalues close to positive real numbers. The results shows that such "copy" maps are ubiquitous in Attention Heads (here 10 out of 12 maps are positive):



(14)

Such Attention Heads may be interpreted as **universally-quantified** ($\forall$) logic formulas. For example when we say "all men are mortal":

$$\forall x.\ \text{Human}(x) \Rightarrow \text{Mortal}(x) \tag{15}$$

any object instantiated as x (eg. "Socrates") would have to be **copied** from the LHS to the RHS.

3 Relation to the biological brain

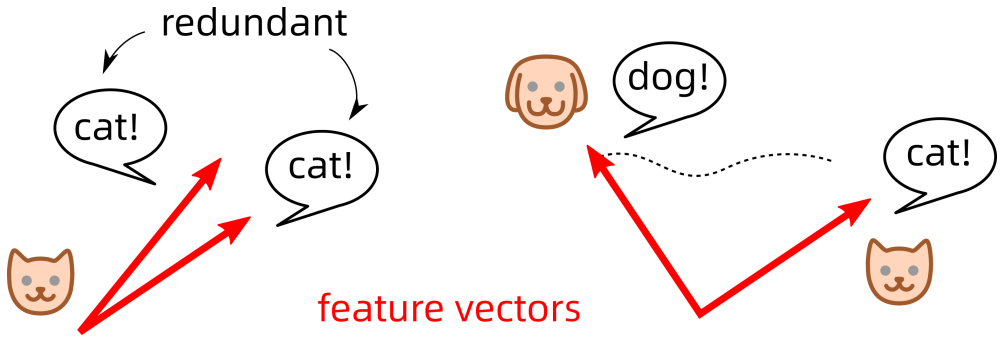
There are two distinct aspects in the brain:

- **Short-term** or Working Memory is the **electric activation** of neuronal populations.
- **Long-term** memory is stored as **synaptic strengths**, established by synaptic formation and strengthening. The transfer from STM to LTM is called **memory consolidation**.

One theory has it that the prefrontal cortex maintains a number of "thoughts" with sub-populations or, perhaps, with **micro-columns**. These activated sub-populations are in competition with each other, through **lateral inhibition**. The thought(s) that win are the thoughts we retain – they "make sense".

3.1 How does symbolic logic emerge in the brain?

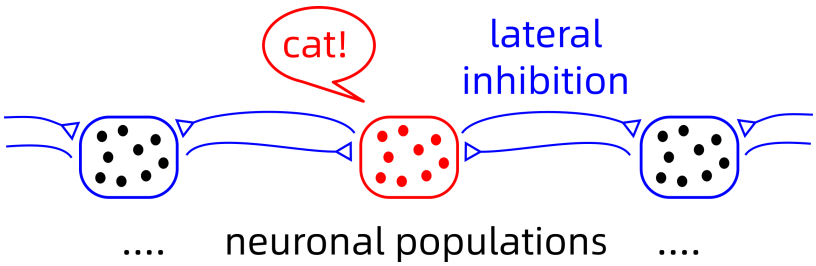
If a room of people see a cat enter the room, one person will say "There's a cat in the room!" but afterwards it would be **redundant** for others to say exactly the same thing. Likewise, in a neural network, if two output features both identify "cat" then they are redundant, a waste of resources. So it is more efficient for one feature vector to move away to a new location in **feature space**:



(16)

The result is the emergence of **disentangled features**. There is now a lot of research papers on this topic; Personally I first learned of this from Marta Garnelo and Murray Shanahan's paper [1]. We can think of this as a first step of **symbolization**, in which objects are recognized by symbols.

In the cortex, neuronal populations are organized into "columns", with **lateral inhibition** among themselves. When one population is activated, it suppresses the activation of nearby populations. This is likely to be the mechanism that enables disentangled features to emerge:

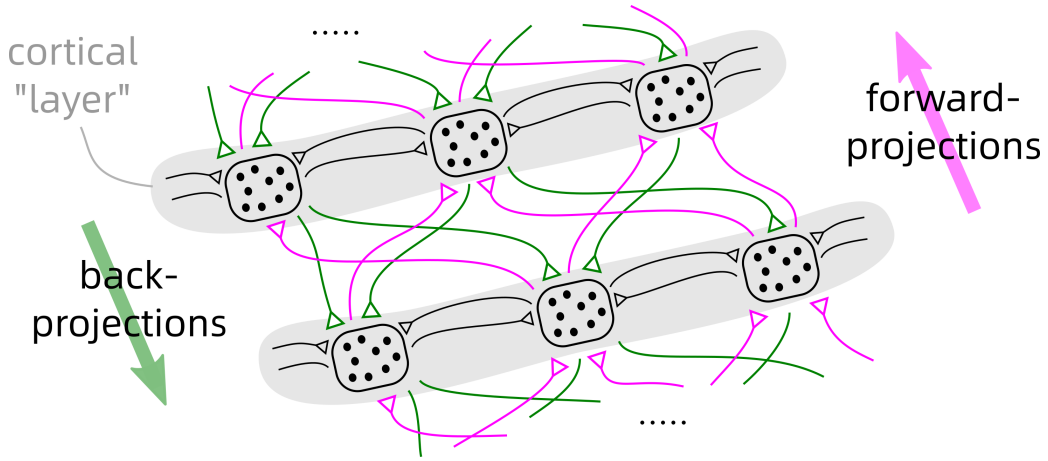


(17)

It is remarkable that the **softmax** in the Transformer / Self-Attention seems to be an abstract implementation of this winner-takes-all **selection** mechanism.

Moreover, the cortex is organized into **layers** with widespread recurrent (ie, forward

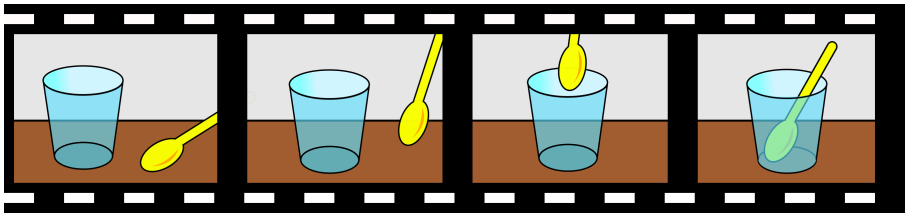
and backward) connections [*] :



(18)

This bi-directional architecture may be applicable to AGI architecture (see also §4 on abductive reasoning), possibly replacing the current uni-directional model of feed-forward networks and the back-propagation algorithm. As is well-known, the brain does not use back-prop. The bi-directional innervation is a very significant brain architectural feature that has not yet been incorporated into current deep learning techniques.

If we consider relations between objects, for example, "spoon inside a glass", this too can emerge out of disentanglement of features, because it is a very **economical** / efficient representation of a complex scene:



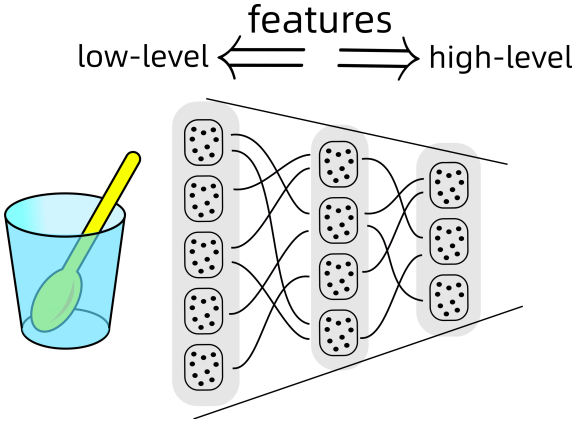
(19)

Every human can recognize this as "putting a spoon into a glass", a symbolic representation. Many researchers may have under-estimated how much the brain uses symbolic reasoning, and my proposal is that AGI can be based entirely on it.

One remaining question is how to represent symbolic data in a "neural" manner. A general form of symbolic data may be as a **tree**. Taking inspiration from the cortex (18), we may perhaps represent the tree / symbolic data as hierarchically

[*] More accurately, there exist two distinct structures: the cortex has a 6-layer structure which has recurrent connections within it; and each cortical area has bi-directional connections to and from other areas (which may have hierarchical relations among themselves). I have sort of glossed over this level of details.

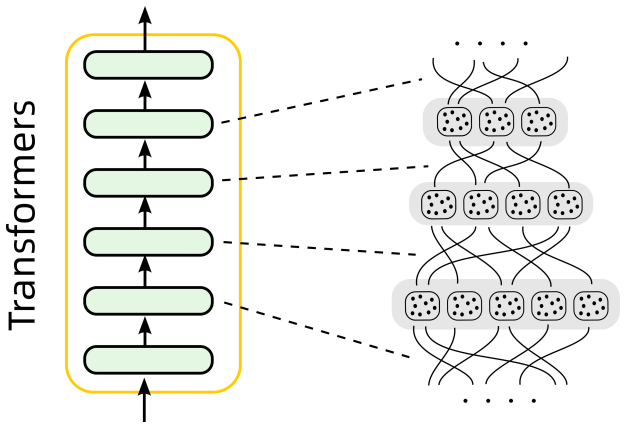
organized neural **feature vectors**:



(20)

Remember that in the Transformer, symbols are organized as **sequences**, for example: "spoon · inside · glass." It may be desirable for AGI to have multiple levels of features, such as "spoon" and "glass" on a lower level, and "inside" on a higher level.

Juxtaposed side by side, the Transformer and the cortex seem to have many similarities:



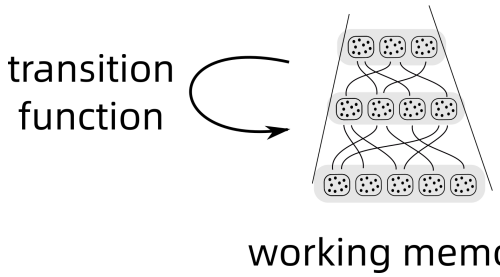
(21)

Softmax corresponds to lateral inhibition. The Transformer has many layers because it **unfolds** along the time axis the training of a recurrent network — part of the reason why the Transformer is very efficient. Each hidden layer of the Transformer can be construed as a "stage" of logical inference:

$$\text{input} \vdash \text{stage}_1 \vdash \text{stage}_2 \vdash .... \vdash \text{output}. \tag{22}$$

Also recall that our reinforcement learning model consists of just the state and its

transition function:



(23)

Based on this understanding, we need to figure out how to design the next version of Transformer and incorporate it into our AGI architecture....

4 Abductive reasoning

Abduction has been relatively neglected in AGI research, which had focused on forward inference. Recently there is a call to study this important aspect.

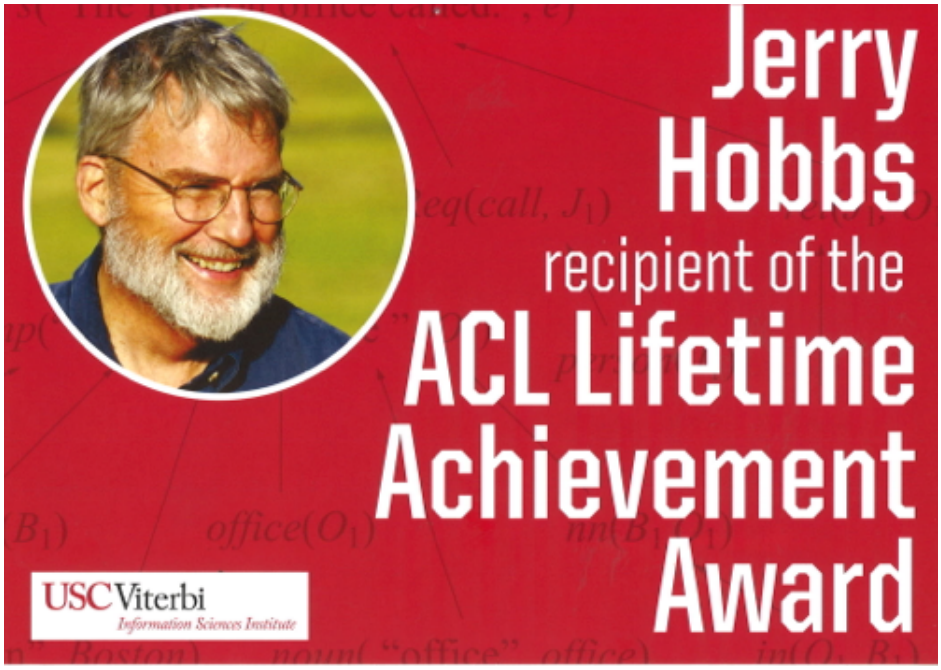
In logic, abduction means finding the **explanation** for some known facts. An explanation E is simply some propositions that imply the known fact F, ie, $E \Rightarrow F$.

For example, why do we think a certain actress, say Marilyn Monroe, is "sexy" [*] ? That's because we recognize she has some features (visual or otherwise, no need to enumerate them explicitly) that we consider sexy. So, $E_1 \wedge E_2 \wedge .... \Rightarrow$ Sexy. Those conditions **imply** she is sexy, and they are the **explanation** for her sexiness.

Why is abduction important? For example, when a waitress says "The Ham Sandwich left a big tip", *Ham Sandwich* here refers to the customer who ordered it (an example of metonymy). The AI knows the plain facts such as that someone ordered a ham sandwich, and then it abduces that the most likely **interpretation** of the phrase "Ham Sandwich" is as the person associated with it. This is the basis of

[*] I use this example because "sexy" is a kind of high-level concept and we apply this concept commonly when we meet other people. Every one can relate to the "reasons" why we think someone is sexy.

Abductive Interpretation of Natural Language proposed by Jerry Hobbs:



(24)

So abductive reasoning is basically just **bidirectional** inference.

When a system has both forward and backward connections, it forms a loop and its dynamics is likely to produce "**resonance**". This harks back to the ART (**Adaptive Resonance Theory**) proposed by Grossberg and Carpenter beginning in the 1980s.

Such resonance behavior can be viewed as the system seeking to minimize an energy, ie, trying to find the "best explanation" to a set of facts.

This is also corroberated by neuroscientific evidence: areas in the cerebral cortex are replete with both forward- as well as **back-projections**, as depicted in diagram (18). We can further abstract this with the following diagram, where F and G are not functions but **optimization constraints**:

$$\begin{array}{c} \vec{y} \\ F \circlearrowright G \\ \vec{x} \end{array} \tag{25}$$

If the input $\vec{x}$ produces the output $\vec{y}$ after some iterations, then it is likely that the output $\vec{y}$ would produce $\vec{x}$ in the **inverse** direction. In other words, we have a

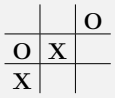
neural mechanism that implements a function f and its inverse f^{-1}. The significance of this (from the **learning** point of view) is that we only need to learn the function f and we get f^{-1} **for free**.

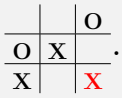
In logic, if forward inference is denoted as $\vdash_{KB}$, where KB is a set of logic rules, then abduction is $(\vdash_{KB})^{-1}$. Abductive interpretation is basically a **constraint-satisfaction** process that uses inference rules in both directions.

5 Dealing with assumptions

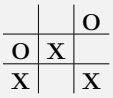
Example of an assumption: "If I play move x now, I can checkmate in 3 moves".

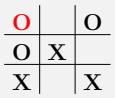
Tic Tac Toe example

Assume the current board is  and it's **X**'s turn to play.

X can do the "double fork" by playing  .

But how can an AGI know (or prove) this?

If the current board is  then a double fork exists. We need a predicate to detect double forks.

We need to reason that even if **O** plays the "blocking" move  , **X** can still win.

We can easily express the conditions for **X**-can-win, but the difficult part is to make the assumption in red, in other words:

$$\text{red-move} \Rightarrow \mathbf{X}\text{-can-win} \tag{26}$$

The difficulty lies in that the LHS is **not true** under the current facts. This conditional statement must be proven by, first, assuming the LHS, and then deriving the RHS. Then the assumption is **discharged** and the conditional statement is proven, via the ($\Rightarrow$ I) rule.

In the old days, in logic-based AI, assumptions are handled with **Truth Maintenance Systems** that keep track of inference traces symbolically. These systems can get quite complicated with the need to track multiple assumptions. For example, when we plan a bank robbery, we need to consider many possible forking scenarios.

Curry-Howard Correspondence

This is the assumption rule (Ax) in proof theory:

$$\Gamma, \phi \vdash \phi \qquad (\text{Ax}) \tag{27}$$

An example instance is:

$$x : \phi,\ y : \psi \vdash x : \phi \qquad (\text{Ax}) \tag{28}$$

Here x is a **λ-variable** of type ϕ. In general, assumptions are λ-variables.

This is the ($\rightarrow$I) "introduction" rule:

$$\frac{\Gamma, x : \phi \vdash M : \psi}{\Gamma \vdash \lambda x.M : \phi \rightarrow \psi} \qquad (\rightarrow I) \tag{29}$$

Thus, **discharging** an assumption (ie, applying the $\rightarrow$I rule) results in a **λ-term**.

Suppose M, N are proofs of $M : \phi \rightarrow \psi$ and $N : \phi$. Then the proof of ψ would be the **application** of M to N, denoted as $@(M, N)$ or simply MN. This is *modus ponens*.

Similarly, for the rule ($\rightarrow$I) in (29), the proof of ψ can be denoted as $\#(x, M)$ or... why not try $\lambda x.M$? Yes, what we get is the lambda-notation. In the words of another author: "The similarity is not intended and not accidental. It is unavoidable." [a] The proof of an implication is a construction, which, according to the Brouwer-Heyting-Kolmogorov interpretation, is a **function**.

[a] Lectures on the Curry-Howard Isomorphism [Sørensen and Urzyczyn 2006], p.56

An AGI needs the ability to place an implication $\phi \rightarrow \psi$ into working memory,

and to prove it using the ($\rightarrow$ I) rule, ie, by making an assumption.

Algorithm: Put the assumption A in Working Memory. Make inferences, marking all conclusions with the prefix $A \rightarrow *$. When enough conclusions are obtained, remove the assumption A from Working Memory.