人工智能的知识表述

甄景贤 general.intelligence@gmail.com

September 13, 2018

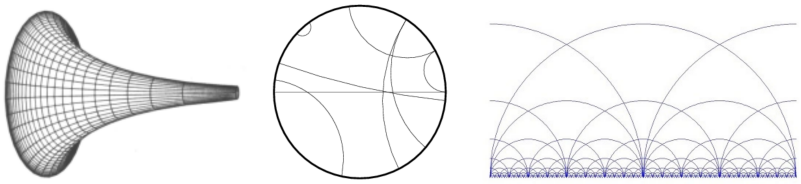
Contents

Abstract

As of now (2018 August), the question of strong AI is no longer whether it is possible or not, but whether one feasible approach is better than another. The tutorial introduces the mathematical theory of knowledge representations, and discusses 3 proposals, respectively based on: A) genetic algorithms; B) neural networks + graphs; C) geometric models.

What is model theory?

For example, **hyperbolic geometry** can be "realized" by the following **models**:

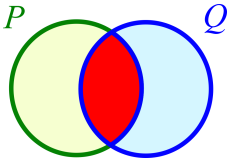


pseudo-sphere Poincaré disc Poincaré half-plane (1)

Models are not unique, there can be many models for a theory.

In mathematical logic, **model theory** studies the **duality** between **syntax** and **models**.

The most classic example is **Stone duality**, also familiar to all of us as "Venn diagrams":

$$P \wedge Q \quad \cong \quad \text{  } \tag{2}$$

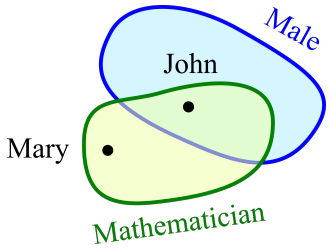
Stone duality* refers to the duality between **Boolean algebras** and **topo-**

*Marshall Stone (1903-1989), American mathematician who contributed to real analysis, functional analysis, topology and the study of Boolean algebras.

logical spaces.

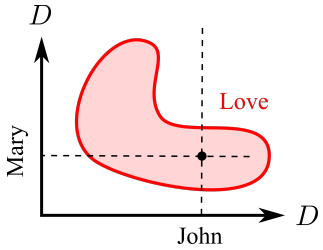
Boolean algebra is the same as **propositional logic**, which concerns only with the truth and falsehood of propositions. For example, P = *"It is raining in New York"*, but we cannot "access" the internal constituents of the propositioon, such as *"rain"* and *"New York"*. The critical obstruction to strong AI is the **lifting** from propositional to first-order logic.

First-order logic can be modeled by **sets** and their **elements**. For example John $\in$ Male, John, Mary $\in$ Mathematician**:**



(3)

Whereas **relations** between first-order **objects** in a domain D are represented by **subsets** of the Cartesian product $D \times D$, eg:



(4)

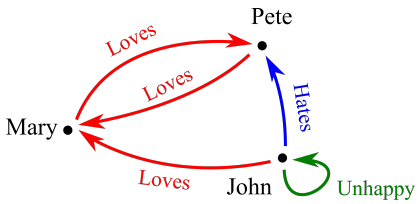
What the fuck's wrong with this line?

For computer science people,

you may be more familiar with **relation graphs**

or **knowledge graphs** such as this one:

What the fuck's wrong with this line?



(5)

This is a **directed multi-graph**, or **quiver**. Quiver is an important structure in algebraic representation theory. The category of quivers $\mathcal{Q}$ is a **topos** (this is introduced in the the basic meaning is it has conditional first-order logic as **model** category [†] .

The above knowledge graph can be simply converted to a collection of **logical children**:

```
Loves(John, Mary)
Loves(Pete, Mary)
Loves(Mary, Pete)
Hates(John, Pete)
Unhappy(John)
```

(6)

So, logic and graph are basically **equivalent**.

If each edge of graph can contain any number of vertices, then there is **hyper-graph**. In other words, each edge of the hypergraph $\in \wp(V)$, V is the vertex set. It can also be said that the hypergraph is the **subset system** of V. For logic, the benefit is: There can be relationships on the relationship. Hypergraph can correspond to the topological **simplicial complex**, and its homology and cohomology can be studied. The Simplicial complex can also correspond to **square-free monomial ideals** one-to-one. Square-free means that the index of x_i can only be 0 or 1. The latter is the scope of the **combina commutative algebra**. For the time being, I don't know if these associations are useful. See [Brown 2013], [Miller and Sturmfels 2005] for details.

A collection of logical expressions is called logical **theory**. A collection of algebraic equations is called **algebraic theory**. For example, you can have

[†]According to [Grilliette 2017], hyper-graph is not topos, multi-graph is not topos, but when they become directed.

the following logical formula ("love is not happy"):

$$\forall x, y.\ \text{Loves}(x, y) \wedge \neg\text{Loves}(y, x) \rightarrow \text{Unhappy}(x) \tag{7}$$

This formula contains universal quantification, so it is not part of the model. Logically, only the collection of **ground sentences** (the expression without variables) can form a model, for example (6) .

Fractal structure of model space

An expression in Logic **theory** can cause many **new** vertices and joins to appear in the model. This is a question of model theory research. In some cases, the model space will have an "infinitely subdivided" fractal structure. For example, each natural number $n \in \mathbb{N}$ has its successor $S(n)$. The existence of this function causes a series of **infinite** vertices in the model space:

$$\bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \ \ldots\ldots \tag{8}$$

If you add this **rule**:

$$\forall n \in \mathbb{N}. \quad S(n) \geq n \tag{9}$$

Then immediately generate an infinite number of relationships:

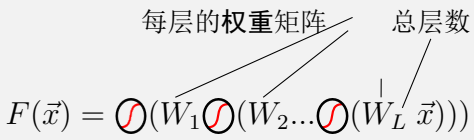
$$\bullet \xleftarrow{\geq} \bullet \xleftarrow{\geq} \bullet \xleftarrow{\geq} \bullet \xleftarrow{\geq} \bullet \xleftarrow{\geq} \bullet \xleftarrow{\geq} \bullet \xleftarrow{\geq} \bullet \ \ldots\ldots \tag{10}$$

Although, in **common-sense intelligence**, it seems that this infinite structure is less common, and more is the structure of "shallow". Incidentally, the knowledge representation of classical logic-based AI is split into two parts: **rules** and **facts**. The former is the formula with $\forall$ **variable**, the latter is ground sentences. Rules are stored in KB and facts are stored in **working memory**. The former is a **theory**, which can be thought of as some **"partial" models**. The reason for saying partial is because it does not represent the entire model. In fact, model is a very large thing that cannot be stored in a physical system. Artificial intelligence or the brain can only store certain theories and parts of models. The key issue of artificial intelligence is how to find a good syntax structure to make theory learning faster and more efficient.

A concise formulation of strong AI

This section describes the mathematical structure of a strong artificial intelligence system in a way that is as concise as possible, even if the mathematician without an AI background can understand it.

Revisit **neural network** like this:



$$F(\vec{x}) = \sigma(W_1\sigma(W_2...\sigma(W_L\vec{x}))) \tag{11}$$

Its **parameter** collection $\Theta = \{W_{i,j}^{\ell}\} \in \mathbb{R}^m$, where $m = \#$ weights. The purpose of **machine learning** is to find optimal Θ subject to an objective function. In other words, machine learning = **optimization**, which is one of the most basic problems in applied mathematics. When training, give a set of data points (F is a neural network):

$$\boxed{\text{training examples}} \quad e \xrightarrow{F} a \quad \boxed{\text{answers}} \tag{12}$$

The error for each answer is ϵ, and the objective function J is the sum of the errors of many iterations.We want to optimize J by calculating the gradient of J for Θ:

$$\nabla_\Theta J := \frac{\partial J}{\partial \Theta} \tag{13}$$

当然这就是著名的 back-propagation 算法，其实即 gradient descent。

强人工智能 的樽颈问题 是**学习算法**的速度 (14)

The feature of the AI learning algorithm is that needs to be optimized on **logical sub**:

从 optimization over $\mathbb{R}$ 过渡到 optimization over $\mathcal{L}$

$$\Theta \in \mathbb{R}^m \quad \rightsquigarrow \quad \Theta \in \mathcal{L} \tag{15}$$

Where $\mathcal{L}$ is some sort of (eg first-order predicate) **logical syntax**, and Θ is a collection of logical children. The solution to this optimization problem Θ^* is an optimal logic theory. The top-level architecture of the system is reinforcement learning, which is dynamic programming. It is a special case of optimization. It can also be called control theory. The **system** it controls is:

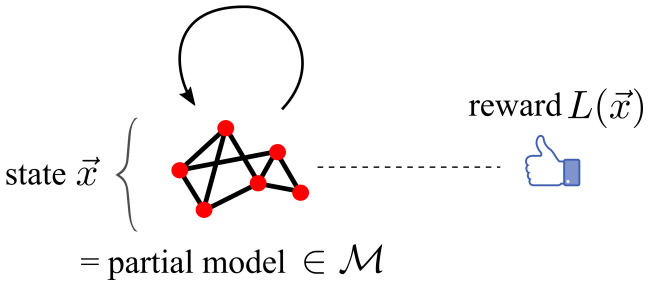
$$\vec{x}_{n+1} = \vec{f}(\vec{x}_n, \vec{u}_n) \tag{16}$$

$\vec{x}$ is the system's **status**, and $\vec{u}$ is called control or action. In AI, $\vec{x}$ is the **location** in the "think space", and $\vec{u}$ is the "thinking" step. We want to control $\vec{u}$ so that the system reaches the maximum value of **rewards** in long running:

$$\boxed{\text{total rewards}} \quad J = \sum_n L(\vec{x}) = \int L dt \tag{17}$$

$L \in \mathbb{R}$ is the **instant reward** at the $\vec{x}$ position. Based on historical analysis mechanics, L is also called **Lagrangian**, unit It's energy (but the sign changes, the latter measures the penalty), but note that $\vec{x}$ is a "think space", unlike physical space. I am writing a differential form to make it easier to remember. The system operates as follows:

$$\left.\begin{array}{ll} \text{model rewriting} & \vec{u} : \mathcal{M} \to \mathcal{M} \\ & \vec{u} : \quad \vec{x} \mapsto \vec{x}' \end{array}\right\} = \Theta$$

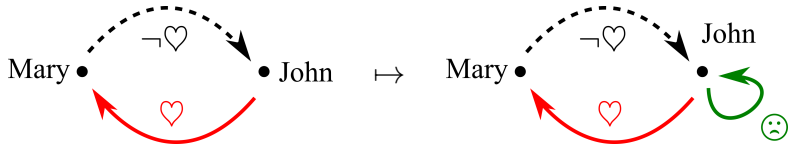


(18)

$\vec{u}$ and $\vec{f}$ coincide, the function is to **rewrite** $\vec{x}$:

$$\vec{f}(\vec{x}, \vec{u}) \equiv \vec{u}(\vec{x}) \tag{19}$$

For example, the logic rule "'失恋 ⇒ 不开心" performs the rewriting of the following sub-graph:



(20)

This is the **state transition** $\vec{u} : \vec{x} \mapsto \vec{x}'$, which can also be regarded as the **logical inference** $\vec{u} : \vec{v} \vdash \vec{x}'$, where $\vec{u}$ is the rewriting function or logic rule. Optimization of J acts on top of Θ (ie $\vec{u}$). Note that $\vec{u} \in$ is a function space or logical formula, which is very different from the traditional optimization over $\mathbb{R}$. In classic AI, $\vec{u}(\vec{x}) = \vec{f}(\vec{x})$ is equivalent to deduction $\vec{x} \vdash \vec{x}'$, or forward-chaining (forward some logical conclusions). In classic AI this work is handled by **logic engine**, which contains two algorithms: **unification** (= rule matching) and **resolution** (= proof search). These operations are not visible in our framework because they are included in $\vec{u}$ or $\vec{f}$.

For experts in applied mathematics, the following theory is quite standard. The definition of Hamiltonian in analytical mechanics is:

$$H = L + \frac{\partial J}{\partial \vec{x}} \vec{f} \tag{21}$$

Similarly, the Hamiltonian of the **discrete** system can be defined as:

$$H = L + J(\vec{f}(\vec{x})) \tag{22}$$

Pontryagin[‡] 的 **极小值原理** 给出最优解的条件是：

$$H^* = \inf_u H \quad \text{or} \quad \nabla_{\vec{u}} H^* := \frac{\partial H^*}{\partial \vec{u}} = 0 \tag{23}$$

可以定义一个作用在 J 上的算子 T：

$$TJ := \inf_u H \tag{24}$$

Then Bellman's optimality condition can be expressed as the following fixed-point form [Bertsekas 2013]:

$$J^* = TJ^* \tag{25}$$

[‡]Lev Pontryagin (1908-1988) Soviet mathematician. Blind since the age of 14, he made major discoveries in a number of fields including algebraic topology and differential topology.

It is said that the Hamilton-Jacobi method leads to the solution of partial differential equations, which is not particularly effective. In practice, the Pontryagin minimum principle is more useful. There is a gradient $\nabla_{\vec{u}}$ in (23), so it would be useful to find a derivative for $\vec{u}$. If the problem is continuous optimization, you can use non-smooth analysis. The minimum requirement is that there is norm (*ie*, Hilbert space or Banach space) on the domain, you can use the method such as proximal gradient. The key issue is: Θ belongs to a domain other than the traditional $\mathbb{R}^n$.

Plan 0: geometric models

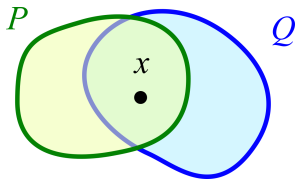
一阶逻辑语法 $\mathcal{L}$ 不是必需的，只需某种将 model **改写** 的能力 (26)

There can be many different variants, for example:

$$\begin{array}{ccc} \text{逻辑语法 } \mathcal{L} & & \text{graph re-writer} \\ \downarrow & \approx & \downarrow \\ \text{partial model } \mathcal{M} & & \text{graph model} \end{array} \tag{27}$$

Consider a model I call **geometric model**, for example:

$$P(x) \wedge Q(x) \quad \cong \quad \tag{28}$$

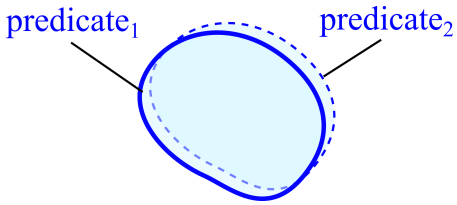


These geometric "**regions**" are easy to do with neural networks. If $\vec{F}$ = neural network, then:

$$\vec{F}(\vec{x}) = 0 \tag{29}$$

定义了一个 hyper-surface of co-dimension 1，它将周围的空间分割为 > 0 和 < 0 两部份。

One effect that you hope to achieve (but perhaps hard to do) is: a logical formula can be deformed into another different expression. This is not possible in classic logic, such as $\text{Love}(x, y)$ and $\text{Like}(x, y)$, although the meaning is close, but from one to the other must be discrete Jumping. As shown below, two approximate regions:



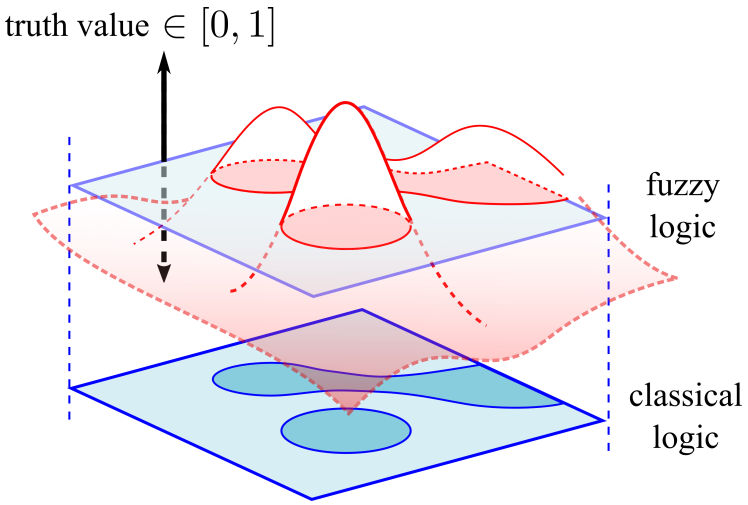
(30)

它们是不是两个 distinct objects 视乎 representation 怎样而决定。

In summary, this program is:

$$\begin{aligned} \text{model } \mathcal{M} &= \text{geometric model of set theory} \\ \text{rewriter} &= \text{functor } \mathcal{M} \to \mathcal{M} \end{aligned} \tag{31}$$

But we need a model of set theory that can be continuously changed, maybe it can be achieved with some kind of fuzzy topology? E.g:

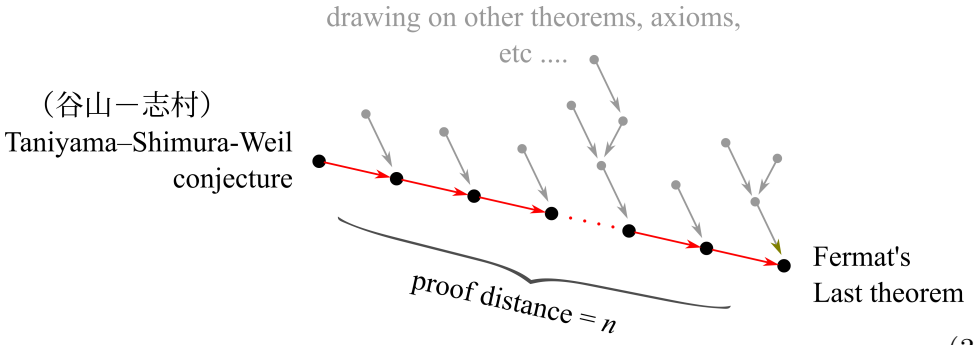


(32)

The boundary of regions that need to be defined is 1 degree higher.

But here's the problem: Logically, the semantic distance between two expressions can be defined as the number of proof steps required to derive from one formula to another (it

also depends on the other within KB (), but according to the halting problem theorem that Turing proved in 1936, this distance is **uncalculable**. [§] In other words, it is impossible in the space of the logical subroutine There is a fully defined semantic metric. Our $\vec{u} = \Theta$ is similar to a logical subroutine. If we can define gradient ∇_Θ on Θ, it seems to be contradictory. Will **fuzzy topology** be able to approximate the semantic metric? This depends on the fuzzy writing model's rewriting function space Θ is not **metrizable** (*cf* [Goubault-Larrecq 2013]). I will study this in more detail in the future.



(33)

The following main analysis of plan A and B (they are already feasible). Plan A uses discrete optimization directly, so no metric space is required. Plan B uses a neural network whose weights $\in \mathbb{R}$ is a contiguous space, but this neural network acts on the graph, which is still a discrete structure.

Plan A: genetic algorithm

放弃梯度下降 (34)

[§]The logic semantic distance is similar to **Kolmogorov complexity** but not exactly the same. Kolmogorov complexity is incomputable but can be approximated [Li and Vitanyi 2008].

$$\begin{aligned} \text{model } \mathcal{M} &= \text{符号逻辑式子} \\ \text{rewriter} &= \text{符号逻辑式子} + \text{经典 AI 逻辑引擎} \end{aligned} \tag{35}$$

GA is very suitable for the **discrete** search space. It is compatible with the logical structure. There is no theoretical obstructions on this route.

First you need a logic-based **rule engine**, which is responsible for forward-chaining, which is completely the scope of classic AI. For example, the classic Soar architecture [Carnegie-Mellon University] is a rule-base engine.

The genetic algorithm's population is composed of individual logical rules, but the winner is not a single rule, but a set of rules (the highest score of N). This is called **cooperative co-evolution**(COCO). Inputs and outputs are logic formulas, which are actually easier to handle. The whole system is still based on reinforcement learning, but you don't need to do RL directly, because those rules are actually **actions**, and the probabilistic strength of each rule is like the Q value in Q-learning. [I have not had time to explore the practical theory of COCO.]

Plan B: neural network / deep learning

Most of the time is to solve the problem of how to implement the classic logic engine with NN, especially the problem of variable substitution. Finally, the crux of the problem is the lack of short-term memory mechanism. The solution is to use graph as **memory system** and use neural network for graph re-writing, which is the graph neural network proposed by Google / DeepMind. This is a "hybrid" architecture.

Neural networks has difficulty handling substitutions

Consider the logic rule that was mentioned in the previous section ("Love is not happy"):

$$\forall x, y.\ x \heartsuit y \wedge \neg y \heartsuit x \to \frownie x \tag{36}$$

The **predecessor** (antecedent) of this rule must be established, and must be a **equal**, twice appearing, b **equal**:

$$a\heartsuit b \wedge \neg b\heartsuit a \tag{37}$$

Also, to produce the correct **postsent** (consequent), you need to pass a **copy** from the previous:

$$a\heartsuit b \wedge \neg b\heartsuit a \rightarrow \text{☹}a \tag{38}$$

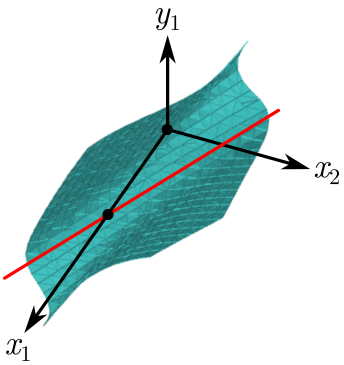
These two actions (**compare** and **copy**) are hard to do with neural networks. But they are the essence of variable substitution and the trouble of predicate logic. In other words, it is difficult to accomplish these two actions in one breath with a monolithic end-to-end neural network:

$$a\heartsuit b \wedge \neg b\heartsuit a \longrightarrow \text{☹}a \tag{39}$$

First consider the copy problem of **poster**. For simplicity, assume that the logical variable z corresponds to some component of the input vector $\vec{x}$, such as x_i. The purpose of Copy is to copy x_i to the y_j position:

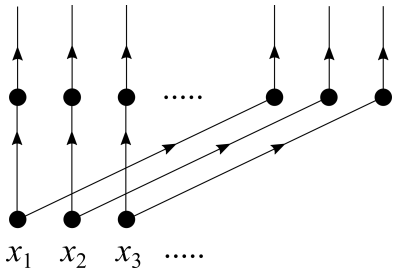
$$\vec{F}: \ (x_1, ..., x_i, ..., x_n) \mapsto (y_1, ..., y_j, ..., y_n) \tag{40}$$

This requires the function surface of the neural network to pass through some diagonal lines, as shown below:



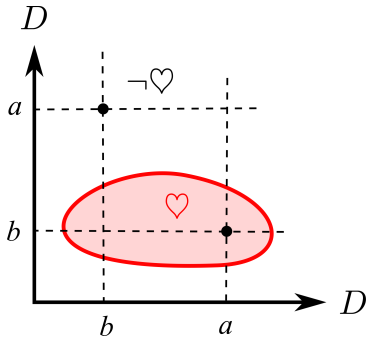
(41)

The following is a simple copier neural network (ownership weight = 1, other weights = 0 not shown):



(42)

A neural network with input dim = n and output dim = $2n$, if fully connected, needs to train $2n^2$ weights. But I haven't had time to test how long it takes to train a multi-layer neural network to learn this action. Second, consider the establishment of **frontware**, a viable **geometric image** is such a [¶]:



$\Rightarrow \odot a$ (43)

$\in$ is easy to solve with a neural network, for example ♡ can be defined as a neural network:

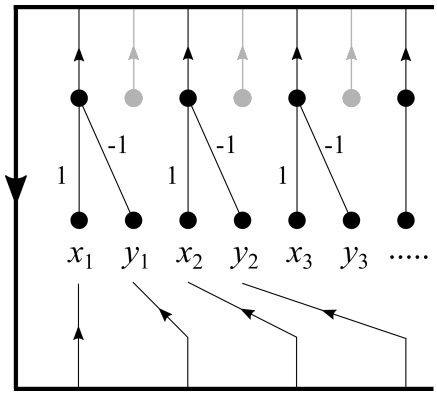
$$\heartsuit(\vec{x}) = \begin{cases} 0 & \vec{x} \notin \text{region} \\ 1 & \vec{x} \in \text{region} \end{cases} \tag{44}$$

但即使这样，仍然馀下一个 pattern matching (comparison) 问题：

$$\begin{array}{rcl} \vec{p}_1 = (\vec{a}, \vec{b}) & \in & \heartsuit \\ \Big\Vert & & \\ \vec{p}_2 = (\vec{b}, \vec{a}) & \in & \neg\heartsuit \end{array} \tag{45}$$

[¶]this is just one of many possible representations, but it seems that any "geometric" form of representations has similar problems. . Unless we consider representations with the characteristics of "procedural"? The following will discuss ....

The following is a simple comparator simulated with the RNN neural network (all = 0 weights are not shown):



(46)

在 iterate n 次之后，最左边的输出 会是 $\vec{x} \stackrel{?}{=} \vec{y}$ 的真假值。假设 输入维数是 $2n$，需要训练 $(2n)^2$ 个 weights。

Conclusion: Based on the above analysis, it seems not very difficult to simulate copier and comparator with NN, but in fact, these "components" are used in conjunction with short-term memory, and the entire architecture is still unknown.

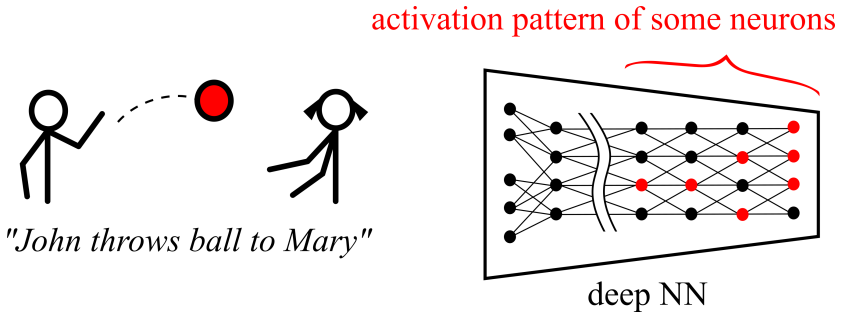
"Distributive" representations

The meaning of **Distributive representation** is: Suppose there is a vector indicating that the output of the neural network has $n = 10$ granule neurons:

$$\vec{x} = (x_1, x_2, ..., x_{10}) \tag{47}$$

In binary, each $x_i \in \{0, 1\}$, $\vec{x}$ can represent the concept of 10 "**one-hot**" respectively. But if you use distributive representation, these 10 bits can express up to $2^n = 1024$ different states/concepts. But in fact, the conjunctions of one-hot features are not different from distributive representations if they are treated as different states. So, the representation of a neural network can be said to be $\mathbb{R}^n$ vector , or as n coordinates of the n-dimensional manifold. For example, *"John throws ball to Mary"* This image, after processing such

as CNN, can get a **distributed knowledge representation**:



(48)

This can be understood as: a "neat" proposition is broken down into a number of small propositions, for example:

$$\text{A 掷球给 B} \Longleftrightarrow \begin{cases} \text{A 手臂挥动} \wedge \\ \text{球离开 A 的手} \wedge \\ \text{球在半空飞} \wedge \\ \ldots \end{cases} \tag{49}$$

These two sides are **logically equivalent**. The details of "The ball is red" cannot appear on the right side, because this detail is not left **implication**. But there can be "the ball is usually round". in other words:

$$\boxed{\text{"neat" proposition}} \quad p \Longleftrightarrow \begin{cases} q_1 \wedge \\ q_2 \wedge \\ \ldots \wedge \\ q_n \end{cases} \quad \boxed{\text{distributed propositions}} \tag{50}$$

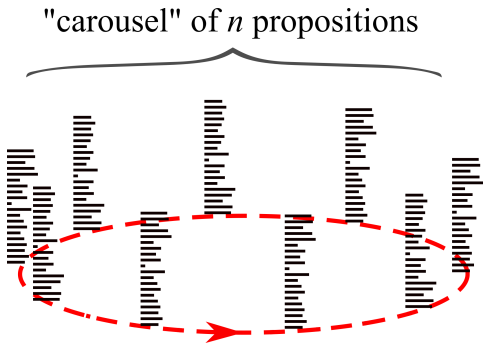
Conclusion: Unknown What is the impact of distributed knowledge representation on AI? For the time being, our architectures are equally applicable to distributive and neat logic, except for the increase in the number of propositions and the part of the "visual nerve".

Neural networks lacks short-term memory mechanism

Consider the image of "White Cat Chasing Black Cat":

(51)

The concept of "cat" needs to appear **twice**, but the feature corresponding to "cat" in the neural network is only **set** (unless there are two duplicate modules that can represent any concept, but it is wasteful) . In other words, the current CNN does not have the ability to "traverse" the field of view; it cannot distinguish and describe the **relationship** between objects. It's hard to imagine how a "monolithic" neural module (such as feed-forward NN or RNN) can do this. It seems necessary to express the proposition as a series of concepts of **time sequence**, some kind of **short-term memory** (**STM**, short-term memory). I was a little surprised to find that the current neural network does not have the mechanism of **short-term memory**, and "short-term" means that the time-scale is shorter than the time when the weights change. For example, I tell you a string of numbers (such as a phone number), you can remember it in the brain, but this mechanism seems to have no research in the current artificial neural network, perhaps some models in computational neuroscience, but for the time being I don't know. Without such an STM, it is difficult to simulate symbolic logic. In other words, strong artificial intelligence cannot be achieved. For example, using NN to implement a **dynamic memory**, when it receives a new element, it will **compare** to other elements in the memory, and it has the **copy** function. For example, the following time series mechanism like "Rotating Trojan" (each ▮▮▮ represents a distributive vector):



(52)

In short, it's obviously cumbersome to simulate STM purely with NN.

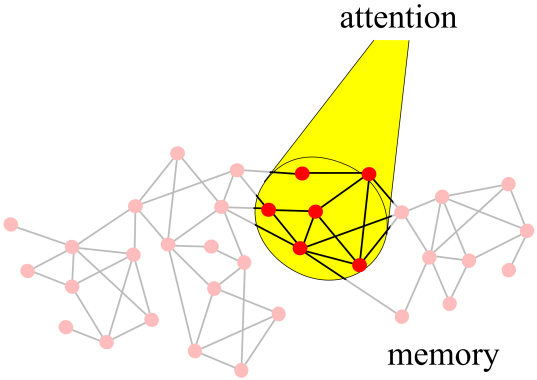
Graph neural networks

用 graph 做记忆体（包括短期和长期记忆） (53)

When adding a new memory unit, the same nodes will be matched into one. In other words, the matching step is solved by the traditional symbolic method, and the problem is left to the neural network.

$$\begin{aligned} \text{model } \mathcal{M} &= \text{graph} \\ \text{rewriter} &= \text{deep NN} = \text{graph neural network} \end{aligned} \tag{54}$$

Many thanks to the graph network paper [Battaglia et al. 2018] published by Google / DeepMind in June 2018, Peter Battaglia and 26 collaborators surveyed the development of graph network. The graph network they proposed is closer to some physical systems such as springs and spheres, rather than the first-order model, but essentially the same. Usually the model is too large, you need to use the attention mechanism to select a fragment of it, and then "present" to handle the neural network:



(55)

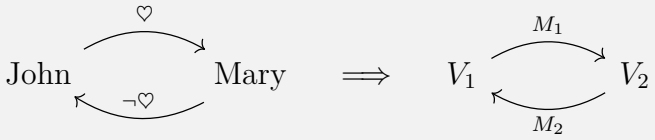
This attention mechanism is somewhat different from the attention in the current deep learning or in the specific details, but basically the same concept. The neural network input $\vec{x}$ requires an embedding like this:

$$\boxed{\text{graph}} \quad \xrightarrow{\text{embed}} \quad (x_1, ..., x_m) \quad \boxed{\text{vector}} \tag{56}$$

But graph is not a "linear" structure [‖], it seems quite difficult to represent the graph structure as a vector (this may be the delay of the graph neural networks) There are reasons for the breakthrough).

[‖] linear means symbolic form, for example, tree can be represented as a linear line

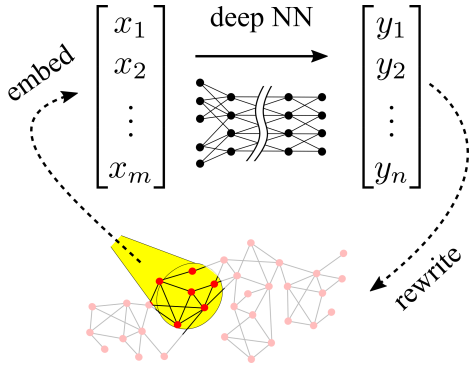
There is **quiver representations** in mathematical representation, which turns vertex into vector space and edge becomes linear transformation between vector spaces. For example:



(57)

Where V_1, V_2 is the vector space, $M_1, M_2 \in GL(\mathbb{R})$ is the matrix. But this is still not a vector. Under the **base transform** of the vector space V_i, the two matrices M may be the same linear transform. Therefore, you need to consider their **invariance**, which is moduli. Representation is concerned with breaking down various M into irreducible components. The Dynkin diagrams that appear in this decomposition are the same as those found in the Lie algebra classification. But if the quiver is not a Dynkin or some extension, then the quiver is "wild" and it's hard to break. Even a simple quiver can be a wild type. Each quiver defines a path algebra whose elements are the path in the quiver, in other words the logical **relationship** and its composition. For the time being, I don't know what quiver representations are for in AI.

The overall operation is like this:



(58)

This deep NN can be either CNN or RNN because they are currently very successful in natural language understanding/translation, but they deal with linear sequence inputs. Or you can split the memory sub-graph into linear elements (that is, individual relationships/propositions), and you can't use

global variable references (in other words, variable matching has been done). Then the variable copying output by NN is also externally processed. In other words, in combination with NN and **graph rewriting** in the "hybrid" way. I still haven't seen a good solution for the embedding details in the picture, but DeepMind / Google's research is very close. Note: Although the parameter space $\Theta \in \mathbb{R}^N$ of NN is **continuous**, the rewriting rules it learns are **discrete** because graph itself is a discrete structure. . Add a point: attention mechanism to traverse memory graph, in other words a graph search algorithm, this part can be combined with NN into the same module (there are many RNN architectures now).

馀下 2 节是一些 数学 背景知识....

Categorical semantics

Categorical semantics is a model theory expressed in category theory. 📖
以下内容主要来自 [Caramello 2018] 这本新书的第一章。更经典的参考书是 [Goldblatt n.d.]. 范畴论最好的入门书当然是「中学生也能看懂的」*Conceptual mathematics* [Lawvere and Schanuel n.d.] 还有 [Lawvere and Rosebrugh 2003].

Different logics can be defined by **proof theory** (which studies *syntactic* rules of deduction):

algebraic logic	no additional rules
Horn logic	finite $\wedge$
regular logic	finite $\wedge$, $\exists$, Frobenius axiom
coherent logic	finite $\wedge$ and $\vee$, $\exists$, distributive axiom, Frobenius axiom
geometric logic	finite $\wedge$, infinitary $\vee$, $\exists$, infinitary distribution axiom, Frobenius axiom
first-order intuitionistic logic	all finitary rules except law of excluded middle
first-order classical logic	all finitary rules

(59)

For example, **algebraic theory** means: It has only one relation $=$, and all axioms are of the form $s = t$. There are also examples of these deduction rules:

$$\boxed{\wedge \text{ rule}} \quad \frac{\Phi \vdash \Psi,\ \Phi \vdash \chi}{\Phi \vdash (\Psi \wedge \chi)} \tag{60}$$

$$\boxed{\exists \text{ double rule}} \quad \frac{\Phi \vdash_{\vec{x},y} \Psi}{\overline{\exists y\, \Phi \vdash_{\vec{x}} \Psi}} \tag{61}$$

$$\boxed{\text{Frobenius axiom}} \quad \Phi \wedge \exists y\, \Psi \vdash_{\vec{x}} \exists y\, (\Phi \wedge \Psi) \tag{62}$$

Tarski's model theory "corresponds" the first-order syntax to the **structure** of the set theory. As a result, different logical syntaxes correspond to different structural categories:

categories with finite products	algebraic logic
Cartesian categories	Cartesian logic
regular categories	regular logic
coherent categories	coherent logic
geometric categories	geometric logic
Heyting categories	first-order intuitionistic logic
Boolean coherent categories	first-order classical logic

(63)

"Geometric" logic means **geometric morphisms**, which can be roughly understood as a mapping between two topois, similar to **continuous maps** between topological spaces. Topos can be understood as a generalization of set theory under the influence of category theory. Each elementary topos[**] There is a **sub-object classifier** Ω. Ω is a special object, such as $\{\top,\ Bot\}$, for **true and false** binary. Use Ω to define sub-objects, which is the concept of **subset** in set theory. For example, "Love" is a **relationship** within $D \times D$, and D is a collection of all "people." You can think of $D \times D$ as full relation, then Love $\subset D \times D$ is its **subset**. In other words, this is why elementary topos can be used as a **model** for relational algebra or first-order logic. (see [Goldblatt n.d.]) "*Lawvere and Tierney's elementary topoi theory* [††] *is the most important event in the history of categorical algebra..... It's not just that they prove these things, but that they dare to believe this. It is*

[**] Elementary is the meaning of "element" in the collection

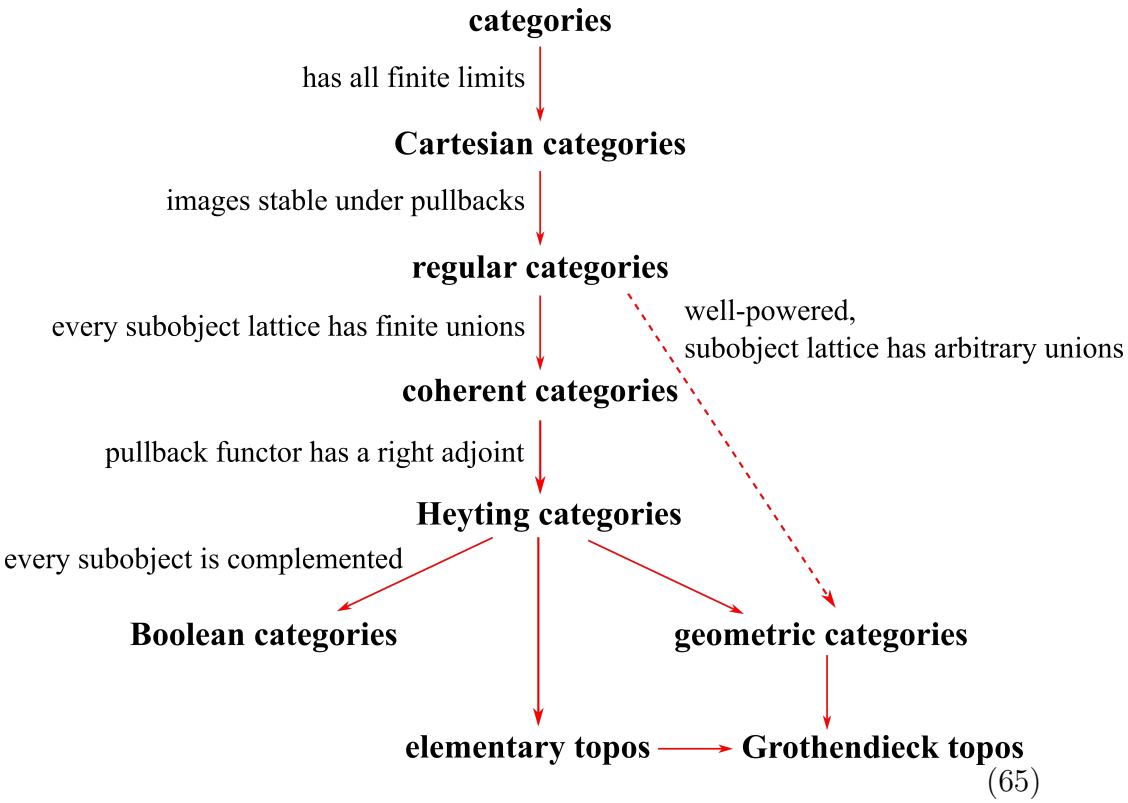
[††] around 1960-70

possible"— Peter Freyd. Some history: Around 1963, the concept of topos came independently from several different sources: Alexander Grothendieck's sheaf theory in algebraic geometry, and F William Lawvere's restatement of set theory with category theory, and Paul Cohen's forcing Theory (the latter is used to solve the **continuity hypothesis**). Sheaf means: Objects defined on some open sets V_i, they are consistent on the overlap $V_i \cap V_j$, which can be "collate". The situation is like some charts in differential geometry. . After World War II, Leray, then Cartan, defined sheaf using the open sets method. Later Lazard defined sheaf with étale, which is the main motivation for the topos theory. For example, a pre-sheaf on a category $\mathcal{A}$ can be defined as a **functor**:

$$\widehat{\mathcal{A}} : \mathcal{A}^{\mathrm{op}} \to \mathbf{Set} \tag{64}$$

That is, a "layer" set-valued function on the category $\mathcal{A}$. It is "functor" so it handles those collating. This functor is contravarient so there is op. Grothendieck Applying sheaf to topology (cohomology), then Jean-Pierre Serre found that it can also be used in algebraic geometry. They and other collaborators wrote the 1623-page masterpiece "SGA IV", which greatly influenced the development of algebraic geometry, resulting in In 1974 Deligne solved the Weyl conjecture. But I am not familiar with algebraic geometry for the time being, so it is not clear what Grothendieck did.... See the book [MacLane and Moerdijk 1992] for details. In the topology space, the complement of an open set U is closed and not necessarily open, so if it is confined within the open sets, the "negation" of U should be defined as "the interior of its complement' '. This causes U's "**double negative**" not necessarily equal to U, in other words, the algebra of open sets follows **intuitionistic logic**, such an algebra is called a **Heyting algebra**. (cf. ibid.) There are two kinds of morphisms between Topoi: **geometric morphisms** and **logical functors**. The former maintains "geometry", the latter maintains a logical type theory, so there is a definition of **elementary topos**. The latter is characterized by its **sub-object classifier** Ω. 以下是根据 [Caramello 2018] Ch.1 整理出来的一张关系图，但我暂时还不太熟悉范畴论的概念，所以也

不完全理解：



(65)

Domain theory

Both λ-calculus and combinatory logic can express any form of **function**. If the domain of the whole function is D and the number of functions by $D \to D$ is $|D^D|$, $|D^D|$ must be greater than $according to Cantor's theorem of set theory. |D|$, even if D is infinite, it is still true. In other words, λ-calculus and combinatory logic are unlikely to have models. This conclusion is very disturbing. But in 1971, the problem was solved by Dana Scott and C Strachey, creating **domain theory**. 以下内容主要来自 [Vickers 1989]，是一本很易懂的书，还有更新和更详尽的 [Goubault-Larrecq 2013].

Scott's solution is to give domain D endow with a **Scott topology** and then only consider $D \to D$'s **continuous function**. The latter is a small number, so it avoids Cantor's theory. ..... 【未完待续】

References

Battaglia et al. (2018). "Relational inductive bias, deep learning, and graph networks". In: URL: https://arxiv.org/pdf/1806.01261.pdf.

Bertsekas (2013). *Abstract dynamic programming*.

Brown (2013). *Discrete structures and their interactions*. CRC Press.

Caramello (2018). *Theories, sites, toposes –- relating and studying mathematical theories through topos-theoretic 'bridges'*.

Goldblatt (n.d.). *Topoi –- the categorical analysis of logic*.

Goubault-Larrecq (2013). *Non-Hausdorff topology and domain theory –- selected topics in point-set topology*. Cambridge new mathematical monographs 22.

Grilliette (2017). "A Functorial Link between Quivers and Hypergraphs". In: URL: https://www.researchgate.net/publication/305787097_A_Functorial_Link_between_Quivers_and_Hypergraphs.

Lawvere and Rosebrugh (2003). *Sets for mathematics*. Cambridge.

Lawvere and Schanuel (n.d.). *Conceptual mathematics*. Cambridge.

Li and Vitanyi (2008). *An introduction to Kolmogorov complexity and its application (3rd edition)*. Springer.

MacLane, Saunders and Ieke Moerdijk (1992). *Sheaves in geometry and logic – a first introduction to topos theory*. Springer.

Miller and Sturmfels (2005). *Combinatorial commutative algebra*. GTM 227.

Vickers (1989). *Topology via logic*.