

人工智能的知识表述

甄景贤 general.intelligence@gmail.com

August 29, 2018

Contents

什么是 model theory?	2
模型空间 的 fractal 结构	4
强人工智能的简单表述	5
Plan 0: geometric models	8
Plan A: genetic algorithm	10
Plan B: neural network / deep learning	11
神经网络 处理 substitutions 的困难	11
「分布式」知识表述	14
神经网络 缺乏短期记忆	15
Graph neural networks	16

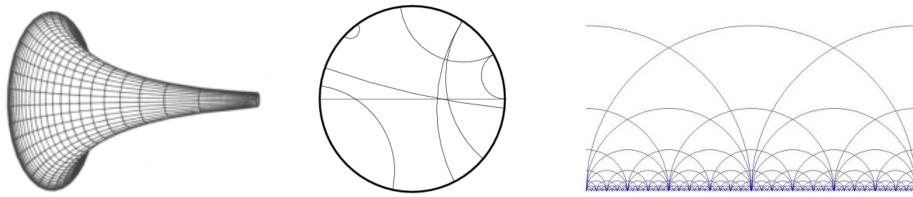
Categorical semantics 19
Domain theory 22

Abstract

目前（2018 年 8 月）强人工智能的发展，问题已经不再是「能不能做到」，而是到了「哪个方案比较好」的地步。本文介绍知识表述的理论，顺带提出两个方案，分别基於：A) 基因算法；B) 深度学习。

什么是 model theory?

举例来说，hyperbolic geometry（双曲几何）可以「实现」为某些 模型：



pseudo-sphere Poincaré disc Poincaré half-plane

模型不是唯一的，可以有很多种。

在数理逻辑中，模型论研究的是 syntax 和 model 之间的对偶。

最经典的例子是 Stone duality，也是大家熟悉的“Venn diagram”：

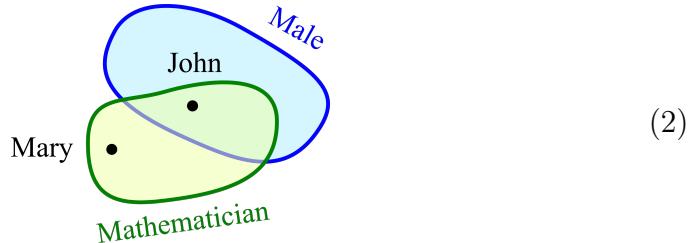
$$P \wedge Q \quad \approx \quad \begin{array}{c} P \\ \cap \\ Q \end{array}$$

Stone duality^{*} 指的是 Boolean algebra 和 topological space 之间的对偶。

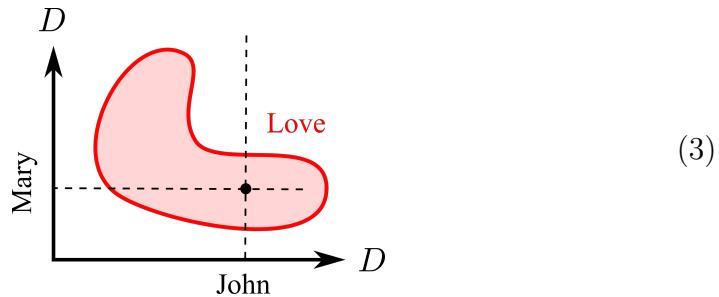
* Marshall Stone (1903-1989), American mathematician who contributed to real analysis, functional analysis, topology and the study of Boolean algebras.

Boolean algebra 即命题逻辑，它只关心命题的真假，例如 $P = \text{「北京正在下雨」}$ ，但它不能“access”命题内部的结构，例如「北京」和「雨水」等成分。强人工智能最关键的障碍 (obstruction)，是命题逻辑到一阶逻辑的跃升 (lifting)。

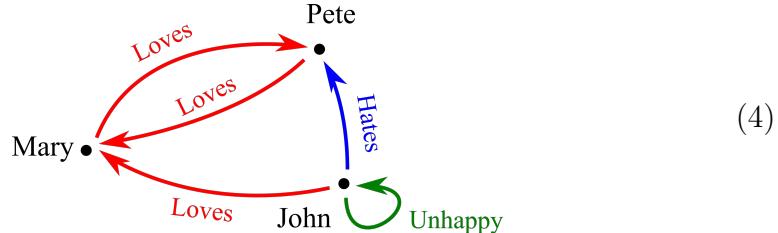
First-order logic 的模型可以用一些集合及其元素组成。例如， $\text{John} \in \text{Male}$, $\text{John}, \text{Mary} \in \text{Mathematician}$:



而 first-order objects (个体)之间的关系是 domain D 的 Cartesian product $D \times D$ 内的一些子集，例如：



对计算系的人来说，更熟识的 model 是以下这种 relation graph 或 knowledge graph:



这是一个 **directed multi-graph**，或者叫 **quiver**。Quivers 是代数表示论 (representation theory) 中的重要结构。Quivers 的范畴 \mathcal{Q} 是一个 **topos** (这在会 § 介绍)，基本意思是 它有条件做 first-order logic 的模型范畴[†]。

[†]根据 [Grilliette 2017] 的说法，hyper-graph 不是 topos，multi-graph 也不是 topos，但当它们变成 directed 则可以。

以上的 knowledge graph 可以简单地转换成 逻辑式子 的集合：

$$\begin{aligned} &\text{Loves(John, Mary)} \\ &\text{Loves(Pete, Mary)} \\ &\text{Loves(Mary, Pete)} \\ &\text{Hates(John, Pete)} \\ &\text{Unhappy(John)} \end{aligned} \tag{5}$$

所以说，逻辑与 graph 基本上是等价的。

如果 graph 的每条边可以包含任意个顶点，则有 **hyper-graph**。换句话说，hypergraph 的每条边 $\in \wp(V)$ ， V 是顶点集。也可以说，hypergraph 就是 V 的 子集系统 (set system)。对逻辑来说，这好处是：关系之上可以有关系。

Hypergraph 可以一一对应於拓扑学上的 **simplicial complex**，可以研究它的 homology 和 cohomology。Simplicial complex 也可以和 **square-free monomial ideals** 一一对应。Square-free 的意思是 x_i 的指数只可以是 0 或 1。后者是 **组合交换代数** (combinatorial commutative algebra) 的研究范围。暂时我不知道这些关联有没有用，详细可参看 [Brown 2013], [Miller and Sturmfels 2005]。

一个逻辑式子的集合叫 **logical theory**. 一个代数等式的集合叫 **algebraic theory**.

例如可以有以下这个逻辑式子 (“失恋则不开心”):

$$\forall x, y. \text{ Loves}(x, y) \wedge \neg \text{Loves}(y, x) \rightarrow \text{Unhappy}(x) \tag{6}$$

这个式子含有 universal quantification，所以不是 model 的一部分。逻辑上来说，只有 **ground sentences** (没有变量的式子) 的集合才可以组成 model，例如 (5)。

模型空间 的 fractal 结构

Logic **theory** 中的一个式子 可以导致 model 中出现很多 新的 顶点和连接。这是 model theory 研究的问题。某些情况下，模型空间 会出现「无限细分」的 fractal 结构。

例如，每一个自然数 $n \in \mathbb{N}$ 都有它的 successor $S(n)$ 。这个函数的存在，导致 model 空间里有一系列 无穷 的顶点：

如果加入这条 法则：

$$\forall n \in \mathbb{N}. \quad S(n) \geq n \quad (8)$$

则立即产生无穷多个关系:

虽然，在 **日常智能** (common-sense intelligence) 中，似乎比较少出现这种无穷的结构，而更多是“shallow”的结构。

附带一提，经典逻辑人工智能 (classical logic-based AI) 的知识表述是分拆成 **rules** 和 **facts** 两部分。前者是带有 \forall 变量的式子，后者是 ground sentences。Rules 储存在 **[KB]** 内，facts 储存在 **working memory** 内。前者是一个 **theory**，后者可以看成是一些 “**partial models**”。说 partial 的原因是因为它不代表整个 model。事实上 model 是非常庞大的东西，不可能储存在物理系统中。人工智能或大脑只能储存某些 theories 和部分的 models。人工智能的关键问题是如何找一种良好的 syntax 结构，令 theory 的学习更快、更有效率。

强人工智能的简单表述

这一节用尽量简练的方式描述强人工智能系统的数学结构，即使没有 AI 背景的数学家也能看懂。

重温一下 神经网络 是这样的：

$$F(\vec{x}) = \bigcirc(W_1 \bigcirc (W_2 \dots \bigcirc (W_L \vec{x})))$$

每层的权重矩阵 总层数

(10)

它的参数集合 $\Theta = \{W_{i,j}^\ell\} \in \mathbb{R}^m$, 其中 $m = \# \text{ weights}$ 。

机器学习 的目的是 寻找 Θ subject to an objective function.
换句话说，机器学习 = **optimization**，这是应用数学的最基本问题之一。

训练时，给定一组 data points (F 是神经网络)：

$$\boxed{\text{training examples}} \quad e \xrightarrow{F} a \quad \boxed{\text{answers}} \quad (11)$$

每个答案的误差是 ϵ , 目标函数 J 是很多次 iterations 的误差之和。我们想令 J 最优化, 方法是计算 J 对于 Θ 的梯度 (gradient):

$$\nabla_{\Theta} J := \frac{\partial J}{\partial \Theta} \quad (12)$$

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 (13)

AI 学习算法的特点是 需要在 逻辑式子 上进行最优化:

$$\text{从 optimization over } \mathbb{R} \quad \Theta \in \mathbb{R}^m \quad \text{过渡到 optimization over } \mathcal{L} \quad \Theta \in \mathcal{L} \quad (14)$$

其中 \mathcal{L} 是某种（例如一阶谓词）逻辑语法， Θ 是一个逻辑式子的集合。这最优化问题的解 Θ^* 是一个 optimal logic theory。

系统的 top-level architecture 是 强化学习，亦即 dynamic programming，是 optimization 的一个特例，也可以叫 control theory，它控制的系统是：

$$\vec{x}_{n+1} = \vec{f}(\vec{x}_n, \vec{u}_n) \quad (15)$$

\vec{x} 是系统的状态, \vec{u} 叫 control 或 action。在 AI 里, \vec{x} 是「思维空间」中的位置, \vec{u} 是「思考」的 steps。我们希望控制 \vec{u} , 令系统在长时间的运行中, 收到的奖励达到最大值:

total rewards

$$J = \sum_n L(\vec{x}) = \int L dt \quad (16)$$

$L \in \mathbb{R}$ 是在 \vec{x} 位置得到的瞬时奖励，基於历史上分析力学的原因， L 也叫作 **Lagrangian**，单位是能量（但正负号改变，后者量度的是惩罚），但注意这里的 \vec{x} 是「思维空间」，不同於物理空间。我附带写上微分形式，以便於记忆。

系统的运行如下：

$$\text{model rewriting } \left. \begin{array}{l} \vec{u} : \mathcal{M} \rightarrow \mathcal{M} \\ \vec{u} : \vec{x} \mapsto \vec{x}' \end{array} \right\} = \Theta$$

state \vec{x} reward $L(\vec{x})$

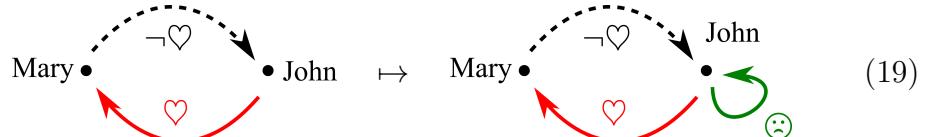
 $= \text{partial model } \in \mathcal{M}$

(17)

\vec{u} 和 \vec{f} 重合，作用是将 \vec{x} 重写：

$$\vec{f}(\vec{x}, \vec{u}) \equiv \vec{u}(\vec{x}) \quad (18)$$

For example, the logic rule “失恋 \Rightarrow 不开心” performs the rewriting of the following sub-graph:



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 作用在 Θ (亦即 \vec{u}) 之上。注意 $\vec{u} \in$ 某种 function space or 逻辑式子，这和传统的 optimization over \mathbb{R} 很不同。

在经典 AI 中， $\vec{u}(\vec{x}) = \vec{f}(\vec{x})$ 的作用相当於 deduction $\vec{x} \vdash \vec{x}'$ ，或者叫 forward-chaining (向前推导出一些逻辑结论)。在经典 AI 中这工作由 **逻辑引擎** 负责，它包含 **unification** (= rule matching) 和 **resolution** (= proof search) 两个算法。在我们的框架中 看不到这些运作，因为它们被包含在 \vec{u} 或 \vec{f} 之内。

对应用数学方面的专家们来说，以下的理论是颇为 standard 的。分析力学中 Hamiltonian 的定义是：

$$H = L + \frac{\partial J}{\partial \vec{x}} \vec{f} \quad (20)$$

类似地，离散系统的 Hamiltonian 可以定义为：

$$H = L + J(\vec{f}(\vec{x})) \quad (21)$$

Pontryagin[†] 的 极小值原理 给出最优解的条件是：

$$H^* = \inf_u H \quad \text{or} \quad \nabla_{\vec{u}} H^* := \frac{\partial H^*}{\partial \vec{u}} = 0 \quad (22)$$

可以定义一个作用在 J 上的算子 T ：

$$TJ := \inf_u H \quad (23)$$

则 Bellman's optimality condition 可以表示为以下的 fixed-point 形式 [Bertsekas 2013]：

$$J^* = TJ^* \quad (24)$$

据说 Hamilton-Jacobi 方法导致解 偏微分方程，不是特别有效，实践中更有用的是 Pontryagin 极小值原理。

在 (22) 式中有梯度 $\nabla_{\vec{u}}$ ，所以如果能对 \vec{u} 求导数是会很有用的。如果问题是 continuous optimization，可以用 non-smooth analysis，其最低要求是 domain 上有 norm (ie, Hilbert space or Banach space)，则可以用 proximal gradient 等方法。

关键问题是： Θ 属於的 domain 不是传统的 \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 (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.

可以有很多不同的变种，例如：

$$\begin{array}{ccc} \text{逻辑语法 } \mathcal{L} & & \text{graph re-writer} \\ \downarrow & \approx & \downarrow \\ \text{partial model } \mathcal{M} & & \text{graph model} \end{array} \quad (26)$$

考虑一种我称为 **geometric model** 的模型，例如：

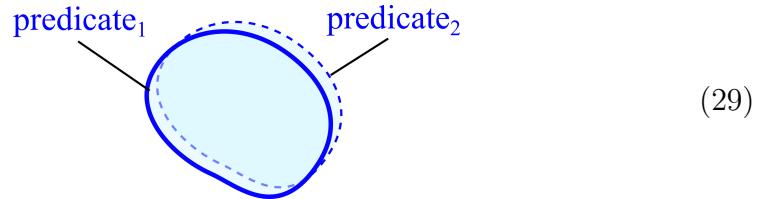
$$P(x) \wedge Q(x) \cong \begin{array}{c} P \\ \cap \\ Q \\ \bullet \end{array} \quad (27)$$

这些 geometric “regions” 很容易用神经网络做到。如果 \vec{F} = 神经网络，则：

$$\vec{F}(\vec{x}) = 0 \quad (28)$$

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。

希望做到（但或许很难做到）的一个效果是：一个逻辑式子能连续地 deform 成另一个不同的式子。在经典逻辑中这是不可能的，例如 $\text{Love}(x, y)$ 和 $\text{Like}(x, y)$ ，虽然意思接近，但从一个变成另一个必需 discrete 的跳跃。如下图，两个近似的 regions :



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

总括来说，这个方案是：

$$\begin{array}{ll} \text{model } \mathcal{M} & = \text{geometric models of set theory} \\ \text{rewriter} & = \text{functor } \mathcal{M} \rightarrow \mathcal{M} \end{array} \quad (30)$$

但我们需要一个可以连续变化的 model of set theory, 或许可以用某种 fuzzy topology 来达成? 这是我未来会专注的方向。

但这里有个问题: 逻辑上, 两个式子之间的 semantic distance 可以定义为由一个式子推导到另一个式子所需的 proof steps 个数 (它还取决于 KB 内的其他式子), 但根据 Turing 在 1936 年证明的 halting problem 定理, 这个 distance 是不可计算的, 它是 Kolmogorov complexity 的一种形式, 它可以被 approximate 但不能完全准确地计算。换句话说, 在逻辑式子的空间上不可能有完全确定的 semantic metric。我们的 $\vec{u} = \Theta$ 类似于逻辑式子, 如果在 Θ 上能定义 gradient ∇_Θ , 似乎有矛盾。会不会 fuzzy topology 能做到 semantic metric 的近似? 这取决于 fuzzy geometric model 的 rewriting function space Θ 是不是 metrizable (cf [Goubault-Larrecq 2013]). 暂时我对 geometric model 的结构仍不熟悉, 所以无法回答这问题。

以下主要分析 plan A 和 B (它们已经是可行的)。Plan A 直接用 discrete optimization, 所以不需要 metric space。Plan B 用神经网络, 它的 weights $\in \mathbb{R}$ 是连续空间, 但这神经网络是作用在 graph 上, 后者仍然是离散结构。

Plan A: genetic algorithm

放弃梯度下降 (31)

$$\begin{aligned} \text{model } \mathcal{M} &= \text{符号逻辑式子} \\ \text{rewriter} &= \text{符号逻辑式子 + 经典 AI 逻辑引擎} \end{aligned} \quad (32)$$

GA 本来就非常适合离散的搜寻空间, 它和逻辑结构很兼容, 在这条路上已经没有理论上的 obstructions.

首先需要一个 logic-based **rule engine**, 它负责 forward-chaining (正向逻辑推导), 这完全是经典 AI 的范围。例如 经典的 Soar architecture [Carnegie-Mellon 大学] 就是一个 rule-base 引擎。

基因算法的 population 是由个别的逻辑 rules 组成, 但 winner 并不是单一条 rule, 而是一整套 rules (最高分的 N 个)。这叫 **cooperative co-evolution** (COCO)。

输入和输出是 logic formulas, 其实更易处理。

整个系统仍然是基於 reinforcement learning 的, 但不需要直接做 RL, 因为那些 rules 其实就是 **actions**, 每条 rule 的 probabilistic strength 就像 Q-learning 中 Q 值的作用。

[我还未有时间 survey COCO 的实践理论。]

Plan B: neural network / deep learning

馀下大部份篇幅都是为了解决如何用 NN 实现 经典逻辑引擎 的问题, 特别是 variable substitution 的问题, 最后发觉问题的癥结在於缺少了 short-term memory 的机制。

解决办法是 用 graph 做记忆系统, 用神经网络做 graph re-writing, 亦即 Google / DeepMind 提出的 graph neural network, 这是一种 “hybrid” architecture.

神经网络 处理 substitutions 的困难

考虑上节讲过的 逻辑 rule (“失恋则不开心”):

$$\forall x, y. \ x \heartsuit y \wedge \neg y \heartsuit x \rightarrow \circledast x \quad (33)$$

这个 rule 的 前件 (antecedent) 要成立, 必须 两次出现的 a 相等、两次出现的 b 相等:

$$a \heartsuit b \wedge \neg b \heartsuit a \quad (34)$$


而且，要产生正确的 **后件** (consequent)，需要从前件中将 a **copy** 过来：

$$a \heartsuit b \wedge \neg b \heartsuit a \rightarrow \oplus a \quad (35)$$

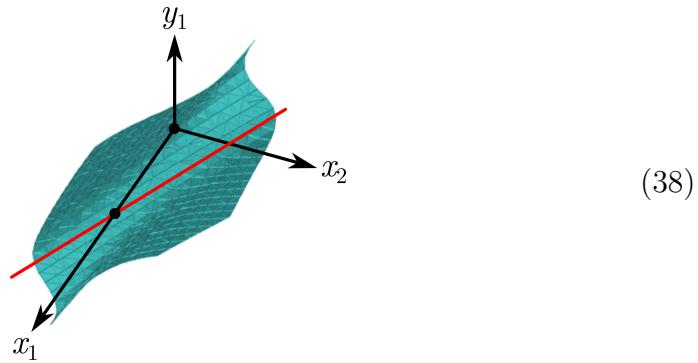
这两个动作（compare 和 copy）都是用神经网络很难做到的。但它们是 variable substitution 的本质，也是谓词逻辑麻烦之处。换句话说，很难用一个 monolithic 的 end-to-end 神经网络一口气完成这两个动作：

$$a \heartsuit b \wedge \neg b \heartsuit a \xrightarrow{\text{ }} \oplus a \quad (36)$$

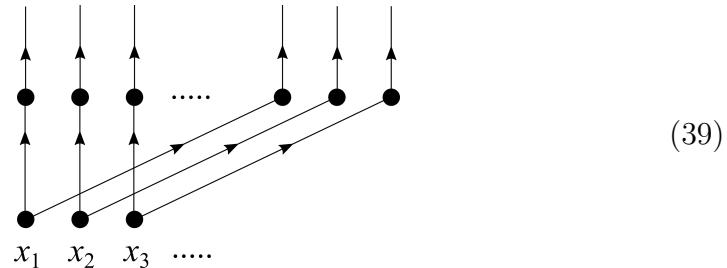
首先考虑**后件**的 copy 问题。为简单起见，假设逻辑 variable z 对应於输入向量 \vec{x} 中的某些分量，例如 x_i . Copy 的作用是将 x_i 抄到 y_j 的位置：

$$\vec{F} : (x_1, \dots, \textcolor{red}{x_i}, \dots, x_n) \xrightarrow{id} (y_1, \dots, \textcolor{red}{y_j}, \dots, y_n) \quad (37)$$

这要求神经网络的函数曲面穿过某些 diagonal 线，如下图：

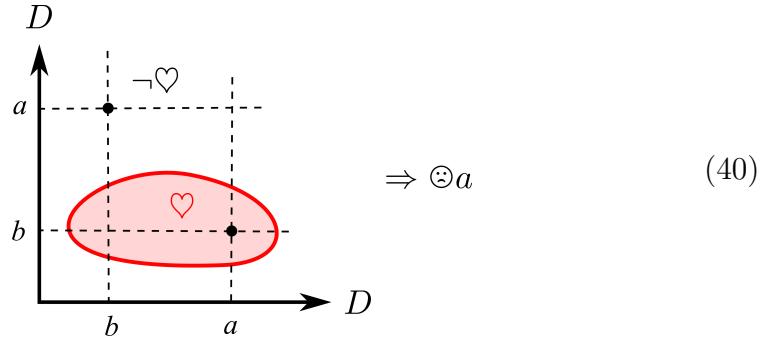


以下是一个简单的 copier 神经网络（所有权重 = 1，其他权重 = 0 没有显示）：



一个输入 $\text{dim} = n$, 输出 $\text{dim} = 2n$ 的神经网络, fully connected 的话需要训练 $2n^2$ 个 weights。但我还未有时间试验一般多层神经网络学习这个动作需要训练多久。

其次, 考虑前件的成立, 一种可行的几何图像是这样的[§]:



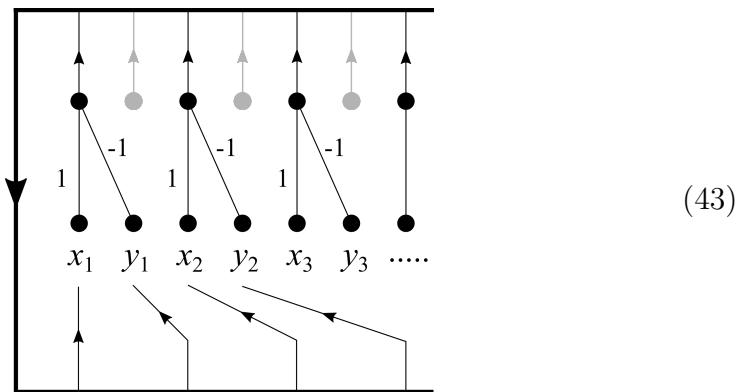
∈ 很容易用神经网络解决, 例如可以定义 \heartsuit 为一个神经网络:

$$\heartsuit(\vec{x}) = \begin{cases} 0 & \vec{x} \notin \text{region} \\ 1 & \vec{x} \in \text{region} \end{cases} \quad (41)$$

但即使这样, 仍然留下一个 pattern matching (comparison) 问题:

$$\begin{aligned} \vec{p}_1 = (\vec{a}, \vec{b}) &\in \heartsuit \\ \cancel{\vec{p}_2 = (\vec{b}, \vec{a})} &\in \neg\heartsuit \end{aligned} \quad (42)$$

以下是一个简单的用 RNN 神经网络 模拟的 comparator (所有 = 0 的权重没有显示):



[§]这只是众多可能的 representations 之一, 但似乎任何「几何」形式的 representations 都有类似问题。除非我们考虑有“procedural”特点的 representations? 以下会讨论....

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。

结论：根据以上分析，用 NN 模拟 copier 和 comparator 似乎不算很难，但实际上还要将这些「元件」配合 short-term memory 使用，整个 architecture 仍然是未知的。

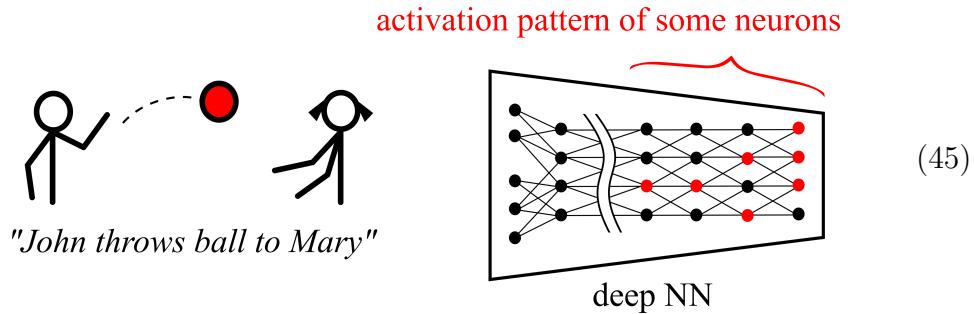
「分布式」知识表述

Distributive representation 的意思是：假设有一个 vector 表示神经网络的输出端有 $n = 10$ 粒神经元：

$$\vec{x} = (x_1, x_2, \dots, x_{10}) \quad (44)$$

用 2 进制，每个 $x_i \in \{0, 1\}$ ，则 \vec{x} 可以分别表示 10 个“one-hot”的概念。但如果用 distributive representation，这 10 个 bits 最多可以表达 $2^n = 1024$ 个不同的状态 / 概念。但其实 one-hot features 的 conjunctions 如果看成是不同的状态，则和 distributive representation 没有区别。所以，神经网络的 representation 本质上可以说是 \mathbb{R}^n vector 而已，或者看成是 n -维流形的 n 个座标。

举例来说，“John throws ball to Mary” 这个图像经过譬如 CNN 的处理后，可以得到一个 分布式知识表述：



这可以理解为：一个“neat”命题被分解为很多个细小的命题，例如：

$$A \text{ 掷球给 } B \iff \left\{ \begin{array}{l} A \text{ 手臂挥动} \wedge \\ \text{球离开 } A \text{ 的手} \wedge \\ \text{球在半空飞} \wedge \\ \dots \end{array} \right. \quad (46)$$

这两边是逻辑等价的。在右边不能出现「球是红色的」这一细节，因为这细节不是左边蕴涵的。但可以有「球通常是圆的」。换句话说：

$$\boxed{\text{"neat" proposition}} \quad p \iff \left\{ \begin{array}{l} q_1 \wedge \\ q_2 \wedge \\ \dots \wedge \\ q_n \end{array} \right. \quad \boxed{\text{distributed propositions}} \quad (47)$$

结论：未知分布式知识表述会给 AI 带来什么影响，暂时我们的 architectures 对 distributive 和 neat logic 同样适用，除了命题数目的增加，与及和「视觉神经」接合的部分。

神经网络 缺乏短期记忆

考虑「白猫追黑猫」这个图像：

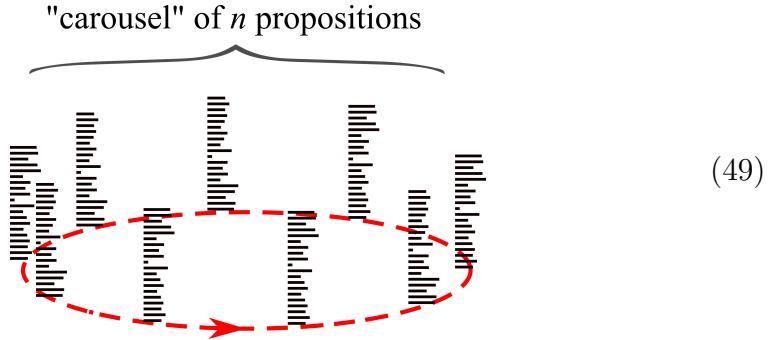


「猫」的概念需要出现两次，但神经网络内对应於「猫」的特徵只有一组（除非有两个重复的可以表示任何概念的 modules，但很浪费）。换句话说，现时的 CNN 没有「巡迴 (traverse)」视野域的能力；它不能辨别和描述物体之间的关系。

很难想像一个“monolithic” neural module（例如 feed-forward NN 或 RNN）怎样可以做到这功能。似乎必须将命题表述成一连串 概念 的 时间序列 (time sequence)，即某种 短期记忆 (STM, short-term memory)。

我有点惊讶地发现，目前 神经网络 没有 短期记忆 的机制，「短期」意思是在 time-scale 上短於 weights 改变的时间。例如我告诉你一串数字（例如电话号码），你可以在脑中记住它，但这个机制在现时人工神经网络里面似乎没有研究，或许在 computational neuroscience 里面有些模型，但暂时我不清楚。缺乏这种 STM，则很难模拟 symbolic logic，换句话说，做不到强人工智能。

例如用 NN 实现一个 动态的记忆体，它接收新来的元素时，会对记忆体中其他元素逐一 比较，而且具备 复制 功能。例如以下这个像「迴转木马」的时间序列机制（每个  代表一支 distributive vector）：



总之，纯粹用 NN 模拟 STM，明显地很麻烦。

Graph neural networks

用 graph 做记忆体（包括短期和长期记忆） (50)

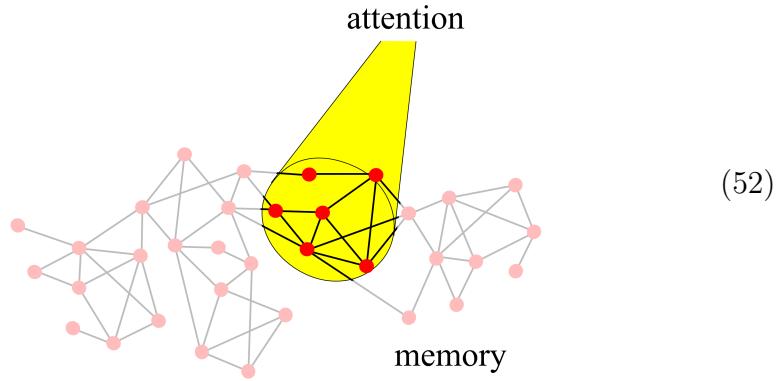
而在增加新记忆单元时，相同的 nodes 会被 match 成一个，换句话说 matching 这步骤用传统 symbolic 方法解决，馀下的问题再交给 神经网络。

$$\begin{aligned} \text{model } \mathcal{M} &= \text{graph} \\ \text{rewriter} &= \text{deep NN} = \text{graph neural network} \end{aligned} \quad (51)$$

很多谢 Google / DeepMind 在 2018 年 6 月发表的 graph network 论文 [Battaglia et al. 2018]，Peter Battaglia 和 26 名合作者 survey 了 graph network 的发展情况。他们提出的 graph network 更接近一些 physical system 例如 弹簧和球体 的系统，而不是 first-order 的模型，但本质上是一样的。

通常 model 太大，需要用 attention mechanism 选取它的一个 fragment，再

“present” 给 神经网络 处理:



这 attention mechanism 和 现时 深度学习 里面的 attention 或者在具体细节上有些分别，但基本上是同一概念。

神经网络的输入 \vec{x} ，需要这样的一个 embedding:

$$\boxed{\text{graph}} \quad \begin{array}{c} \text{graph} \\ \text{---} \\ \text{---} \end{array} \xrightarrow{\text{embed}} (x_1, \dots, x_m) \quad \boxed{\text{vector}}$$

(53)

但 graph 并不是一个「线性」的结构 \ddagger ，将 graph 结构表示成一支 vector 似乎颇难（这或许是 graph neural networks 迟迟未有突破的原因）。

数学表示论里面有 **quiver representations**，它将 vertex 变成 vector space，edge 变成 linear transformation between vector spaces. 例如：

$$\begin{array}{ccc} \text{John} & \xrightleftharpoons[\text{-}\heartsuit]{\heartsuit} & \text{Mary} \\ & \Rightarrow & \\ V_1 & \xrightleftharpoons[\text{-}M_2]{M_1} & V_2 \end{array} \quad (54)$$

其中 V_1, V_2 是向量空间， $M_1, M_2 \in GL(\mathbb{R})$ 是 矩阵。但这仍然不是一 支向量。

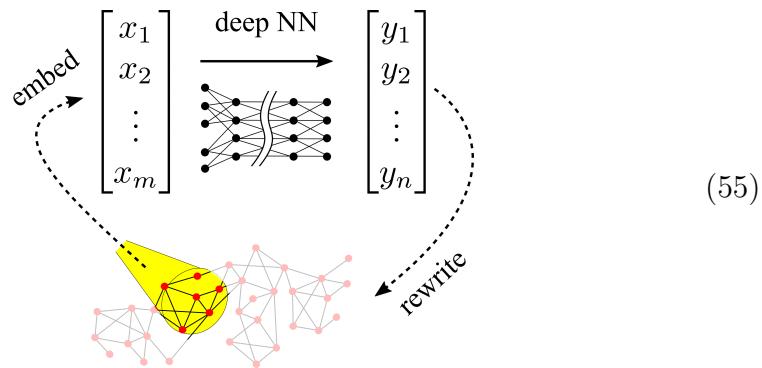
在 向量空间 V_i 的 基底变换下，两个矩阵 M 可能是同一个线性变 换。故需要考虑它们的 不变性，亦即 moduli。表示论 关心的是 将各

\ddagger 线性是指符号形式上，例如 tree 可以表示成线性的一行

种 M 分解成不可约成分。这分解里出现的 Dynkin diagrams 和 Lie algebra 分类时出现的一样。但如果 quiver 不是 Dynkin 或某些扩充，则这 quiver 是“wild”的，很难分解。即使很简单的 quiver 也可以是 wild type。

每个 quiver 定义一个 path algebra，它的元素是 quiver 里的 path，换句话说即是逻辑上的 **关系** 及其 compositions。暂时不知道 quiver representations 在 AI 里有什么用。

整体运作是这样的：



这 deep NN 可以是 CNN 或 RNN，因为它们现时在自然语言理解 / 翻译方面很成功，但它们处理的是线性的 sequence 输入。或者可以将 memory sub-graph 分拆成线性的元素（亦即个别的 **关系 / 命题**），而且里面不可以全局地引用变量（换句话说，已经进行了 variable matching 处理）。然后 NN 输出的 variable copying 也是 externally 处理的。换句话说，用“hybrid”的方式，结合 NN 和 **graph rewriting**。图中那 embedding 细节目前我仍未见过良好的解决办法，但 DeepMind / Google 他们的研究已经很接近了。

注意：虽然 NN 的参数空间 $\Theta \in \mathbb{R}^N$ 是 **连续的**，但它 learn 出来的 rewriting rules 却是 **离散的**，因为 graph 本身是个离散结构。

补充一点：attention mechanism 要 traverse memory graph，换句话说是一种 graph search algorithm，这部分可以和 NN 结合成同一个 module（现时有很多 RNN architectures 就是这样）。

余下 2 节是一些 数学 背景知识....

Categorical semantics

Categorical semantics 是用 category theory 表达的 model 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 1984, 2006]. 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 1991, 1997, 2009] 还有 [Lawvere and Rosebrugh 2003].

不同的 logics 可以透过 **proof theory** (它研究的是 *syntactic* rules of deduction) 定义：

algebraic logic	no additional rules	(56)
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	

举例来说，**algebraic theory** 的意思是：它只有一个 relation $=$ ，而所有 axioms 都是 $s = t$ 这种形式。

还有这些 deduction rules 的例子：

$$\boxed{\wedge \text{ rule}} \quad \frac{\Phi \vdash \Psi, \Phi \vdash \chi}{\Phi \vdash (\Psi \wedge \chi)} \quad (57)$$

$$\boxed{\exists \text{ double rule}} \quad \frac{\Phi \vdash_{\vec{x}, y} \Psi}{\exists y \Phi \vdash_{\vec{x}} \Psi} \quad (58)$$

$$\boxed{\text{Frobenius axiom}} \quad \Phi \wedge \exists y \Psi \vdash_{\vec{x}} \exists y (\Phi \wedge \Psi) \quad (59)$$

Tarski 的模型论 将 first-order syntax 「对应」到 集合论 的 结构 (structures) 上。由此推广，不同的 逻辑 syntax 对应於不同的 结构范畴：

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

(60)

“Geometric” logic 的意思来自 **geometric morphisms**，它可以粗略地理解为两个 topoi 之间的映射，类似於 **continuous maps** between topological spaces。

Topos 可以理解为 set theory 受范畴论影响下的一种推广。每个 elementary topos[†] 有一个 **sub-object classifier** Ω 。 Ω 是一个特殊的 object，例如 $\{\top, \perp\}$ ，代表 真假 二值。用 Ω 可以定义 sub-objects 亦即集合论中的 子集 概念。举例来说，“Love” 是一个在 $D \times D$ 内的 关系， D 是所有「人」的集合。可以将 $D \times D$ 看成是 full relation，则 $\text{Love} \subset D \times D$ 是它的子集。换句话说，这是 elementary topos 可以用来做 relation algebra 或 first-order logic 的 模型 的原因。（参见 [Goldblatt 1984, 2006]）

「Lawvere 和 Tierney 发展的 *elementary topoi* 理论^{**}，是 *categorical algebra* 历史上最重要的事件..... 这不只是他们证明了这些东西，而是他们敢於相信这是可能的」 — Peter Freyd.

一些历史：在 1963 年左右，topos 的概念独立地来自几个不同的发源地：Alexander Grothendieck 在代数几何方面发展的 sheaf theory，和 F William

[†]Elementary 是集合中「元素」的意思

^{**}around 1960-70

Lawvere 用范畴论重新表述集合论，还有 Paul Cohen 的 forcing 理论（后者用来解决 连续统假设）。Sheaf 的意思是：在一些 open sets V_i 上定义的物体，它们在 overlap $V_i \cap V_j$ 上是吻合的，即可以“collate”，情形就像微分几何里一些 charts 拼合成 atlas。二战后，Leray，接著 Cartan，用 open sets 的方法定义了 sheaf。其后 Lazard 用 étale 定义 sheaf，后者是 topos 理论的主要动机。例如，一个范畴 \mathcal{A} 上的 pre-sheaf 可以定义为一个 **functor**：

$$\hat{\mathcal{A}} : \mathcal{A}^{\text{op}} \rightarrow \mathbf{Set} \quad (61)$$

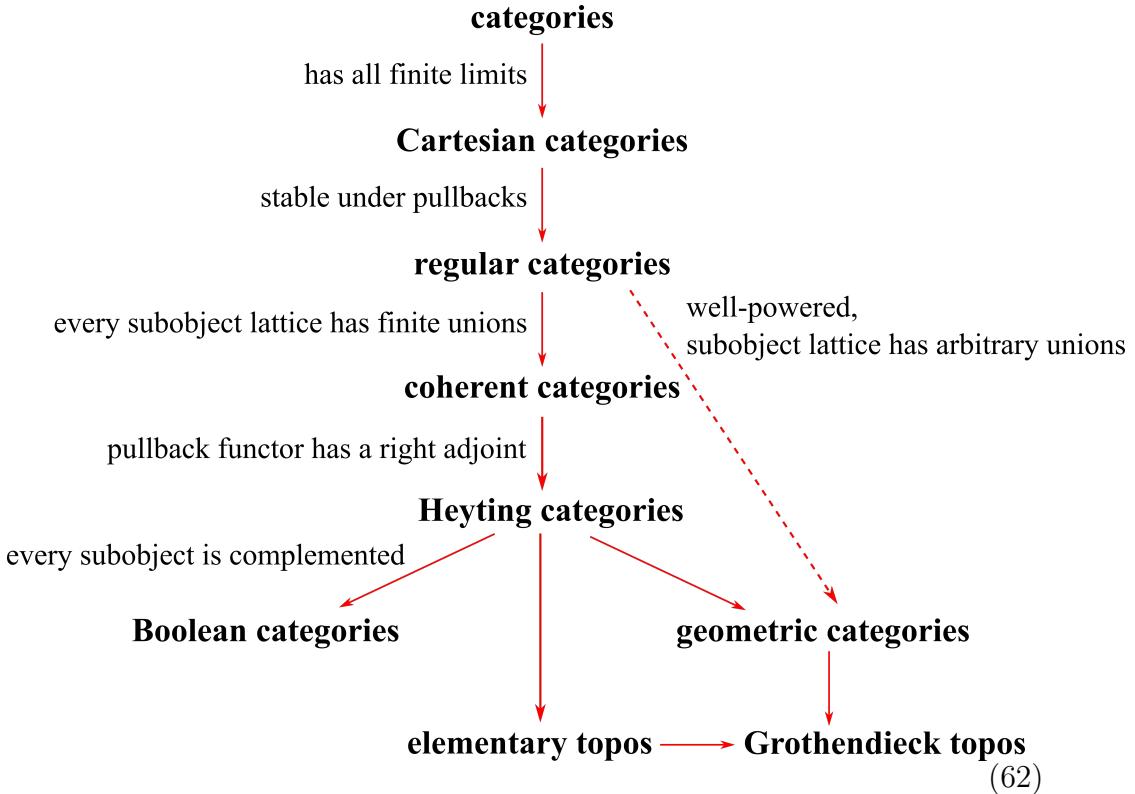
亦即是 范畴 \mathcal{A} 上的一「层」set-valued functions. 它是“functor”所以处理了那些 collating. 这个 functor 是 contravariant 所以有 op. Grothendieck 将 sheaf 应用在 topology (cohomology) 上，而后 Jean-Pierre Serre 发现它也可以用在代数几何上，他们和其他合作者 写了 1623 页的巨著《SGA IV》，重大影响了代数几何的发展，导致 1974 年 Deligne 解决了 Weyl 猜想。但我暂时不熟悉代数几何，所以不太清楚 Grothendieck 他们做了什么.... 详细可参看 [MacLane and Moerdijk 1992] 一书。

在拓樸空间上，一个 open set U 的 complement 是 closed 而且未必 open，所以如果局限在 open sets 之内，则 U 的“negation”应该定义为“the interior of its complement”. 这导致 U 的「双重否定」不一定等於 U ，换句话说，the algebra of open sets follows intuitionistic logic, such an algebra is called a **Heyting algebra**. (参考书同上)

Topoi 之间有两种 morphisms: **geometric morphisms** 和 **logical functors**. 前者保持「几何结构」，后者保持逻辑上的 type theory，所以有 **elementary topos** 的定义。后者的特点是它有 **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：



Domain theory

λ -calculus 和 combinatory logic 都是可以表达任意 **函数** 的形式。如果全体函数的 domain 是 D , 而由 $D \rightarrow D$ 的函数的个数是 $|D^D|$, 则根据集合论的 Cantor's theorem, $|D^D|$ 必定大於 $|D|$, 即使 D 是无穷依然成立。换句话说, λ -calculus 和 combinatory logic 不可能有 models。这结论是非常令人不安的。但在 1971 年, 这个问题被 Dana Scott 和 C Strachey 解决了, 开创了 **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 的解决办法是给 domain D endow with a **Scott topology**, 然后只考虑 $D \rightarrow D$ 的连续函数。后者的数量较少, 所以避开了 Cantor 勃论。

..... 【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*. CRC Press.
- Brown (2013). *Discrete structures and their interactions*. CRC Press.
- Caramello (2018). *Theories, sites, toposes -- relating and studying mathematical theories through topos-theoretic ‘bridges’*.
- Goldblatt (1984, 2006). *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.
- Henrot and Pierre (2018). *Shape variation and optimization*. EMS tracts in mathematics 28.
- Lawvere and Rosebrugh (2003). *Sets for mathematics*. Cambridge.
- Lawvere and Schanuel (1991, 1997, 2009). *Conceptual mathematics*. Cambridge.
- 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*.