人工智能的知识表述

甄景贤

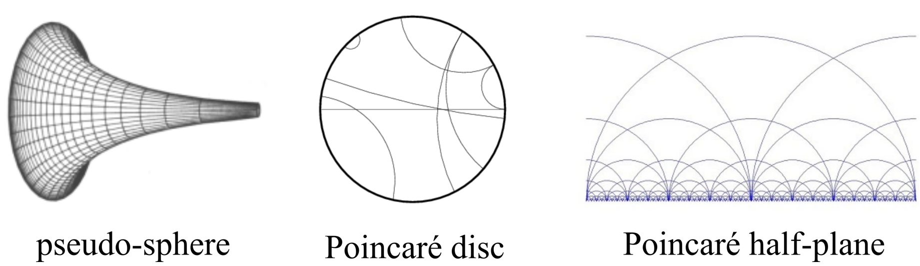
August 22, 2018

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。

1　什么是 model theory？

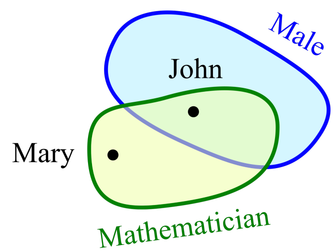
举例来说，hyperbolic geometry（双曲几何）可以「实现」为某些 **模型**：



模型不是唯一的，可以有很多种。

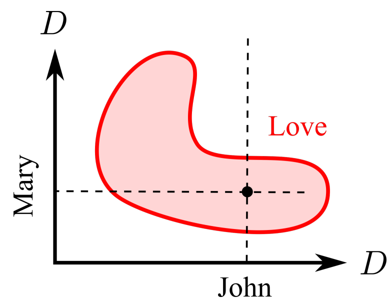
在数理逻辑中，**模型论** 研究的是 syntax / **theory** 和 **model** 之间的 **对偶**。

First-order logic 的 模型 可以用一些 **集合** 及其 **元素** 组成。例如，
John $\in$ Male,   John, Mary $\in$ Mathematician：



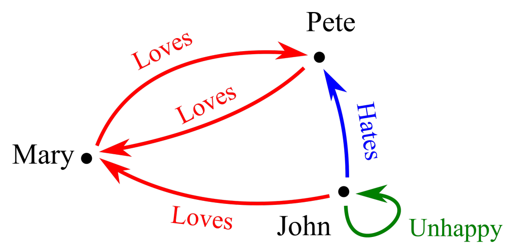
(1)

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：



(2)

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



(3)

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 §8 介绍），基本意思是 <u>它有条件做 first-order logic 的**模型**范畴</u>* 。

*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：

```
Loves(John, Mary)
Loves(Pete, Mary)
Loves(Mary, Pete)
Hates(John, Pete)
Unhappy(John)
```
(4)

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"）：

$$\forall x, y.\ \text{Loves}(x, y) \wedge \neg\text{Loves}(y, x) \rightarrow \text{Unhappy}(x) \tag{5}$$

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 (4) 。

2 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：

$$\bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \ ..... \tag{6}$$

如果加入这条 **法则**：

$$\forall n \in \mathbb{N}. \quad S(n) \geq n \tag{7}$$

则立即产生无穷多个关系：

$$\bullet \xleftarrow{\geq} \bullet \xleftarrow{\geq} \bullet \xleftarrow{\geq} \bullet \xleftarrow{\geq} \bullet \xleftarrow{\geq} \bullet \xleftarrow{\geq} \bullet \xleftarrow{\geq} \bullet \ ..... \tag{8}$$

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。

3 强人工智能

神经网络：

每层的**权重**矩阵　　总层数

$$F(\vec{x}) = \sigma(W_1 \sigma(W_2 ... \sigma(W_L \ \vec{x}))) \tag{9}$$

它的 **参数** 集合 $\Theta = \{W^{\ell}_{i,j}\} \in \mathbb{R}^m$，其中 $m = \#$ weights。

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

强人工智能 的樽颈问题 是**学习算法**的速度

训练时，给定一组 data points（F 是神经网络）：

$$\boxed{\text{training examples}} \quad e \xrightarrow{F} a \quad \boxed{\text{answers}} \tag{10}$$

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)：

$$\nabla_\Theta J := \frac{\partial J}{\partial \Theta} \tag{11}$$

当然这就是著名的 back-propagation 算法。

人工智能的学习算法要求从 optimization over $\mathbb{R}$ **过渡**到 optimization over $\mathcal{L}$（**逻辑式子**上的最优化）：

$$\Theta \in \mathbb{R}^m \quad \rightsquigarrow \quad \Theta \in \mathcal{L} \tag{12}$$

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。

这时，objective function 需要用一个 **逻辑引擎** evaluate，它包含 **unification** 和 **resolution** 两个算法。

训练的过程类似 (10)：

$$\boxed{\text{training examples}} \quad e \vdash_{\Theta} a \quad \boxed{\text{answers}} \tag{13}$$

但现在 $\vdash$ 是根据 Θ 的 **逻辑推导**，e 和 a 是逻辑式子的集合。给定一些前提，可以推出很多结论，其中有些得到奖励或惩罚。在 **强化学习** (reinforcement learning，亦即 dynamtic programming) 的框架下，训练是根据 **奖励** 而不是 **误差**，和上面略有不同。

有几个可能的解决方案：

(plan A) **Genetic algorithm**. 放弃梯度下降，GA 本来就非常适合**离散**的搜寻空间，它和逻辑结构很兼容，在这条路线上已经没有理论上的 obstructions. 见 §4。

(plan B) **Neural network / deep learning**. 从 §5 开始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 DeepMind 提出的 graph neural network，这是一种 “hybrid” architecture. 详见 §7。

(plan C) 放弃一阶逻辑。考虑到以下两者的相似性：

$$\begin{array}{ccc} \text{逻辑语法} & & \text{神经网络 graph re-writer} \\ \big\downarrow & \approx & \big\downarrow \\ \text{partial model} & & \text{graph model} \end{array} \tag{14}$$

或许会想到：其实一阶逻辑语法是不需要的，只需要某种将 model **改写**的能力。当然，如果改写的能力太弱，则会 “throw the baby out with the water”，亦即丧失了一阶逻辑的**泛化**能力。同时，要从 optimization 的角度考虑，什么形式的改写才最有利於速度。模型的 representation 也是一个考虑因素：它应该有某种良好的 **metric**，这 metric 近似 semantic distance，而不是纯粹基於 logic syntax。事实上，完美的 semantic metric 是**不可计算的**，因它是 Kolmogorov complexity. 本篇文章主要分析 plan A 和 B，plan C 是笔者未来会专注的方向。

4 Plan A: 协同进化算法 (COCO)

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。]

5 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”）：

$$\forall x, y.\ x \heartsuit y \wedge \neg y \heartsuit x \rightarrow \frownie x \tag{15}$$

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**：

$$a \heartsuit b \wedge \neg b \heartsuit a \tag{16}$$

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

$$a \heartsuit b \wedge \neg b \heartsuit a \rightarrow \frownie a \tag{17}$$

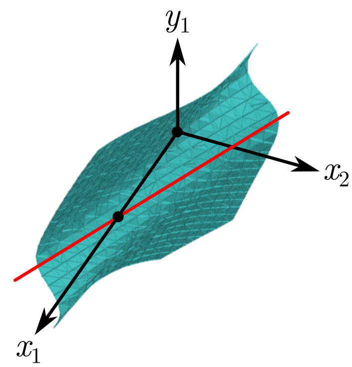
这两个动作（**compare** 和 **copy**）都是用神经网络很难做到的。但它们是 variable substitution 的本质，也是 谓词逻辑 麻烦之处。换句话说，很难用一个 monolithic 的 end-to-end 神经网络 一口气完成这两个动作：

$$a \heartsuit b \wedge \neg b \heartsuit a \longrightarrow \frownie a \tag{18}$$

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：

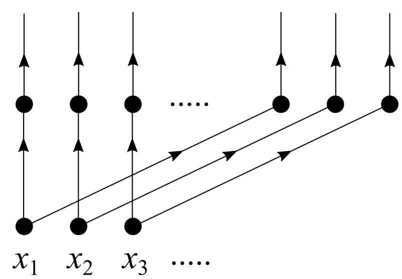
$$\vec{F}:\ (x_1, ..., x_i, ..., x_n) \mapsto (y_1, ..., y_j, ..., y_n) \tag{19}$$

这要求神经网络的函数曲面穿过某些 diagonal 线，如下图：



(20)

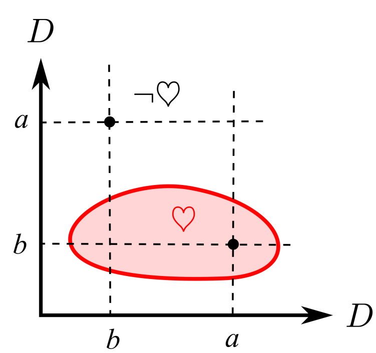
以下是一个简单的 copier 神经网络（所有权重 $=1$，其他权重 $=0$ 没有显示）：



(21)

一个输入 dim $=n$，输出 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 †：



$\Rightarrow$ ☹a (22)

†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....

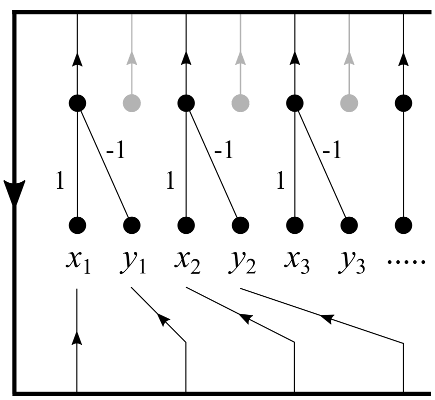
$\in$ 很容易用神经网络解决，例如可以定义 $\heartsuit$ 为一个神经网络：

$$\heartsuit(\vec{x}) = \begin{cases} 0 & \vec{x} \notin \text{region} \\ 1 & \vec{x} \in \text{region} \end{cases} \tag{23}$$

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

$$\begin{aligned} \vec{p}_1 = (\vec{a}, \vec{b}) \quad &\in \quad \heartsuit \\ \vec{p}_2 = (\vec{b}, \vec{a}) \quad &\in \quad \neg\heartsuit \end{aligned} \tag{24}$$

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）：



(25)

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。

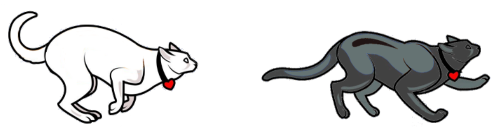
6 神经网络的 知识表述

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：

$$\vec{x} = (x_1, x_2, ..., x_{10}) \tag{26}$$

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。

考虑「白猫追黑猫」这个图像：

 (27)

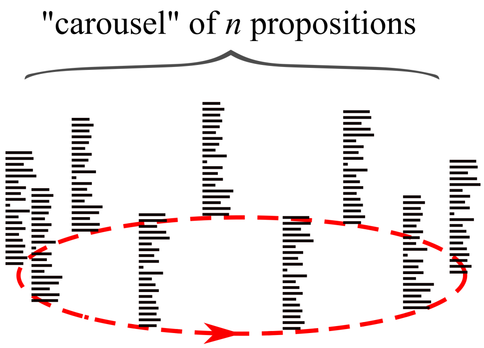
「猫」的概念需要出现 **两次**，但神经网络内对应於「猫」的特徵只有 **一组**（除非有两个重复的可以表示任何概念的 modules，但很浪费）。换句话说，现时的 CNN 没有「巡迴 (traverse)」视野域的能力；<u>它不能辨别和描述物体之间的**关系**</u>。

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）：



(28)

总之，纯粹用 NN 模拟 STM，明显地很麻烦。

用神经网络解决强人工智能的最大障碍是 从命题逻辑到一阶逻辑的 **跃升**，特别是 substitution 运算，需要有 short-term memory 机制

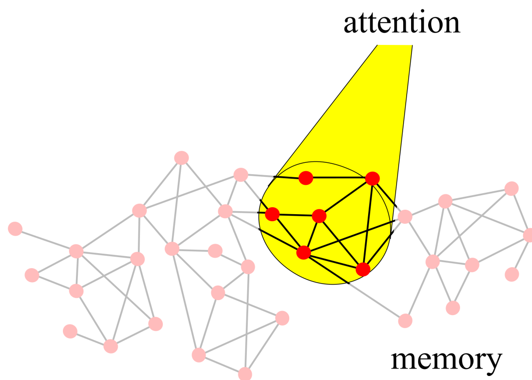
7 Plan B: Graph neural networks

重点是：用 graph 做记忆体（包括短期和长期记忆），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。

很多谢 DeepMind 在 2018 年 6 月发表的 graph network 论文 [Battaglia et al. 2018]，他们 survey 了 graph network 的发展情况。但他们提出的 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。

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：



(29)

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：

$$\boxed{graph}$$

$$\xhookrightarrow{\text{embed}} (x_1, ..., x_n) \quad \boxed{vector} \tag{30}$$

但 graph 并不是一个「线性」的结构 [‡]，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：

$$\begin{tikzcd} \text{John} \arrow[r, bend left, "\heartsuit"] & \text{Mary} \arrow[l, bend left, "\neg\heartsuit"] \end{tikzcd} \quad \Longrightarrow \quad \begin{tikzcd} V_1 \arrow[r, bend left, "M_1"] & V_2 \arrow[l, bend left, "M_2"] \end{tikzcd} \tag{31}$$

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

[‡]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。

解决办法：现时 在 自然语言理解 方面 最强的深度学习模型，据我所知是基於 CNN 或 RNN 的模型，而它们处理的是线性的 sequence 输入。所以要将 memory graph 分拆成线性的元素（亦即个别的 关系 / 命题），而且里面不可以用 global variable references（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**。

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）。

馀下 2 节是一些 数学 背景知识....

8 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].

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	
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	(32)
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：

$$\boxed{\wedge \text{ rule}} \quad \frac{\Phi \vdash \Psi, \; \Phi \vdash \chi}{\Phi \vdash (\Psi \wedge \chi)} \tag{33}$$

$$\boxed{\exists \text{ double rule}} \quad \frac{\Phi \vdash_{\vec{x},y} \Psi}{\exists y \, \Phi \vdash_{\vec{x}} \Psi} \tag{34}$$

$$\boxed{\text{Frobenius axiom}} \quad \Phi \wedge \exists y \, \Psi \vdash_{\vec{x}} \exists y \, (\Phi \wedge \Psi) \tag{35}$$

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	(36)
geometric categories	geometric logic	
Heyting categories	first-order intuitionistic logic	
Boolean coherent categories	first-order classical logic	

"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 $\{\top, \bot\}$，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 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]）

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 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**:

$$\mathcal{A}^{\mathrm{op}} \to \mathbf{Set} \tag{37}$$

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。

[§]Elementary 是集合中「元素」的意思

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：

$$
\begin{tikzcd}
& \textbf{categories} \arrow[d, "\text{has all finite limits}"'] & \\
& \textbf{Cartesian categories} \arrow[d, "\text{stable under pullbacks}"'] & \\
& \textbf{regular categories} \arrow[d, "\text{every subobject lattice has finite unions}"'] \arrow[dddr, dashed, "\substack{\text{well-powered,} \\ \text{subobject lattice has arbitrary unions}}"] & \\
& \textbf{coherent categories} \arrow[d, "\text{pullback functor has a right adjoint}"'] & \\
& \textbf{Heyting categories} \arrow[dl, "\text{every subobject is complemented}"'] \arrow[dd] \arrow[dr] & \\
\textbf{Boolean categories} & & \textbf{geometric categories} \arrow[d] \\
& \textbf{elementary topos} \arrow[r] & \textbf{Grothendieck topos}
\end{tikzcd}
\tag{38}
$$

9 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 $D \to D$ 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 的解决办法是在 domain D 上引入 **Scott topology**，然后只考虑 $D \to D$ 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`.

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`.

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*.