人工智能的知识表述

甄景贤 general.intelligence@gmail.com

October 20, 2018

Contents

Abstract

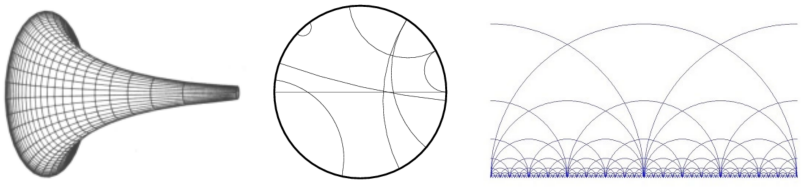
At the top level, an AI can be construed as a dynamical system moving in "cognitive space", trying to minimize a loss function (or maximize a reward). The control of such a system follows the Hamilton-Jacobi-Bellman equation. This framework of reinforcement learning does not offer sufficient structure to solve the AI problem effectively, but it provides a perspective from which subsequent development is possible.

The salient feature of human intelligence is reasoning ability, which is distilled as the theory of logic. We need to impose this logical structure on the AI system (ie, inductive bias). Topos theory provides the perfect abstraction tool to do this. Recently, Google / DeepMind proposed a reasoning architecture based on a neural network acting upon a graph memory. With our abstraction we may be able to improve upon the graph model with simpler and perhaps more efficient "geometric models". The categorical perspective makes it easier to see the problem's structure.

The abstract formulation of AI may lead to better variations. In this paper we look at 3 approaches, based on: A) genetic algorithm; B) neural networks; C) geometric models. This paper is written in a tutorial style as the mathematical background may be unfamiliar to many AI practitioners or researchers.

什么是 model theory?

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



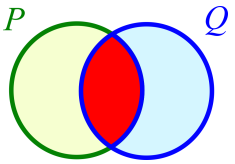
pseudo-sphere　Poincaré disc　Poincaré half-plane

(1)

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"：

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

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

Boolean algebra 即**命题逻辑**，它只关心命题的 真假，例如 P =「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)。

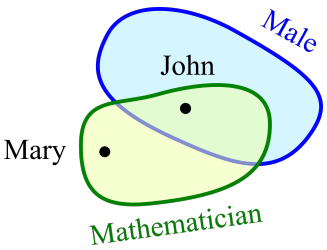
Roughly speaking, classical logic consists of 2 key notions:

- **propositions** are things that can be assigned **truth values**
- each proposition is a **relation** between **objects**

By an informal argument, any idea that is *expressible* in natural language, must be structurally essentially equivalent to predicate logic, which we know from the study of linguistics. So there is no *concievable* way for us to invent an alternative logic that is drastically different from classical logic, with the exception of relatively simple syntactic transformations.

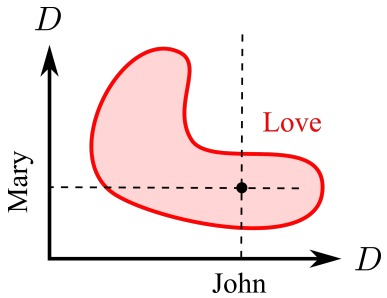
Alan Turing (1912-1954), being very ahead of his time, **circumcribed** the problem of AI by formulating the form of all computable functions, ie, Turing machines, later shown to be equivalent to λ-calculus and combinatory logic. Figure (3) shows the family of major logical structures.

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：



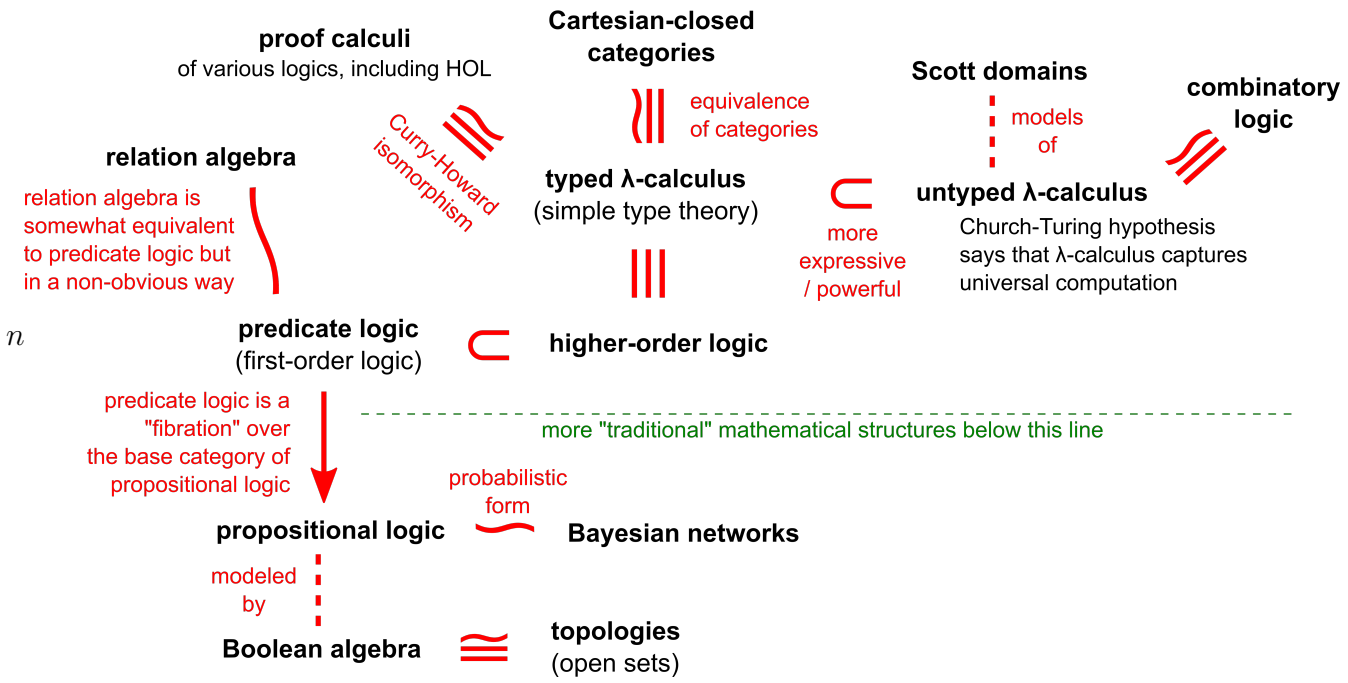
(4)

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：



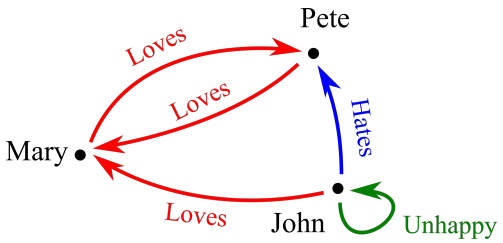
(5)

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



(3)

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



(6)

这是一个 **directed multi-graph**，或者叫 **quiver**。Quiver 是 代数表示论 (representation theory) 中的重要结构。Quivers 的范畴 $\mathscr{Q}$ 是一个 **topos**（这会在下文介绍），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。

Note also that the above graph is somewhat misleading because, strictly speaking, in a topos, **objects** are **types** and **morphisms** are **terms**. So "love" should be a map from the set of people to itself. In our graph this is broken down into individual relations.

Perhaps it is most important to understand that the morphism $A \to B$ in a topos is supposed to mean $A \subset B$ or $A \in B$ in set theory. For example John $\in$ Men and (John, Mary) $\in$ Love.

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)
```

(7)

所以说，逻辑与 graph 基本上是**等价**的。

Hypergraphs and simplicial complexes

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”）：

$$\forall x, y.\ \text{Loves}(x, y) \wedge \neg\text{Loves}(y, x) \to \text{Unhappy}(x) \tag{8}$$

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 (7) 。

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：

$$\bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \ldots\ldots \tag{9}$$

如果加入这条 **法则**：

$$\forall n \in \mathbb{N}. \quad S(n) \geq n \tag{10}$$

则立即产生无穷多个关系：

$$\bullet \xleftarrow{\geq} \bullet \xleftarrow{\geq} \bullet \xleftarrow{\geq} \bullet \xleftarrow{\geq} \bullet \xleftarrow{\geq} \bullet \xleftarrow{\geq} \bullet \xleftarrow{\geq} \bullet \ldots\ldots \tag{11}$$

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。

Partial models in an AI system

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。

Brief review of neural networks

In short, a neural network is a **parametrized** function with **universal approximation** ability.

重温一下 **神经网络** 是这样的：

$$F(\vec{x}) = \sigma(W_1\sigma(W_2...\sigma(W_L\,\vec{x}))) \tag{12}$$

（每层的**权重**矩阵 / 总层数）

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**，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）：

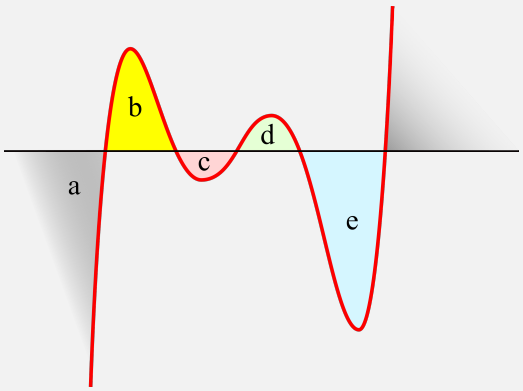
$$\boxed{\text{training examples}} \quad e \xrightarrow{F} a \quad \boxed{\text{answers}} \tag{13}$$

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)：

$$\nabla_\Theta J := \frac{\partial J}{\partial \Theta} \tag{14}$$

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

If the sigmoid function σ is replaced by a polynomial function π , the overall network function would be a composite polynomial whose **degree** is the product of the degrees of every layer. In other words, the total degree grows **exponentially** as #(layers) grows. As the polynomial degree is the number of **zero-crossings**, it can be regarded as the number of possible **classifications** discernable by the network. For example, a quintic polynomial can distinguish up to 5 classes:



(15)

In other words, the classifying power of NN increases exponentially as #(layers).

Consider an NN whose inputs and outputs are **binarized** and of the same size, ie, it maps $\{0,1\}^n \to \{0,1\}^n$. The number of such distinct discrete functions is $(2^n)^{2^n} = 2^{n2^n}$, this number is **doubly exponential** in n, and is so large that in practice it could be regarded as ∞. Yet an NN is able to approximate this function space using $L \cdot n^2$ weights. The **hierarchical** organization of weights is what gives the NN its *"unreasonable effectiveness"*.

强人工智能的樽颈问题是学习算法的速度。

AI 学习算法的特点是 需要在 **逻辑式子** 上进行最优化：

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

$$\Theta \in \mathbb{R}^n \quad \rightsquigarrow \quad \Theta \in \mathscr{L} \tag{16}$$

其中 $\mathscr{L}$ 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：

$$\vec{x}_{t+1} = \vec{f}(\vec{x}_t, \vec{u}_t) \tag{17}$$

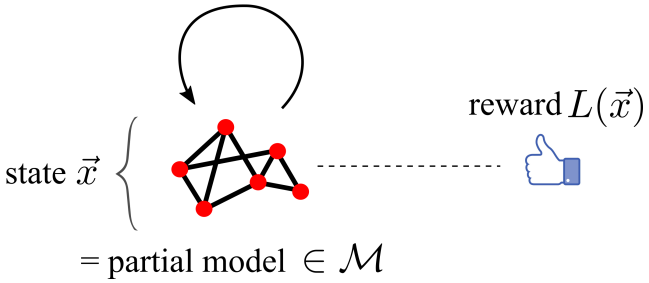
$\vec{x}$ 是系统的**状态**，$\vec{u}$ 叫 control 或 action。在 AI 里，$\vec{x}$ 是「思维空间」中的**位置**，$\vec{u}$ 是「思考」的 steps。我们希望控制 $\vec{u}$，令系统在长时间的运行中，收到的**奖励**达到最大值：

$$\boxed{\text{total rewards}} \quad J = \sum_t L(\vec{x}) = \int L dt \tag{18}$$

$L \in \mathbb{R}$ 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：

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

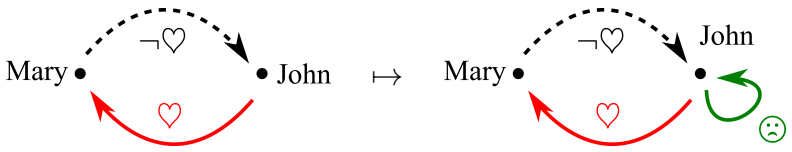


(19)

$\vec{u}$ 和 $\vec{f}$ 重合，作用是将 $\vec{x}$ **重写**：

$$\vec{f}(\vec{x}, \vec{u}) \equiv \vec{u}(\vec{x}) \tag{20}$$

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



(21)

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。

Hamilton-Jacobi-Bellman equation

对应用数学方面的专家们来说，以下的理论是颇为 standard 的。它纯粹设定 AI 系统在强化学习的框架下，除此以外没有其他实质内容。

分析力学中 Hamiltonian 的定义是：

$$H = L + \frac{\partial J}{\partial \vec{x}} \vec{f} \tag{22}$$

类似地，**离散**系统的 Hamiltonian 可以定义为：

$$H = L + J(\vec{f}(\vec{x})) \tag{23}$$

Pontryagin[†] 的 **极小值原理** 给出最优解的条件是：

$$H^* = \inf_u H \quad \text{or} \quad \nabla_{\vec{u}} H^* := \frac{\partial H^*}{\partial \vec{u}} = 0 \tag{24}$$

可以定义一个作用在 J 上的算子 T：

$$TJ := \inf_u H \tag{25}$$

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

$$J^* = TJ^* \tag{26}$$

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 (24) 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。

One question is: why hasn't the keyword "**symplectic**" appeared in current AI literature on reinforcement learning, while all existing techniques are **statistical** in nature? I suspect that it may be due to either: 1) the time-step being discrete; or 2) the dimensionality of the state space being too high (though this dimensionality may not be *intrinsic* to the problem, *ie*, it may be possible to embed into lower dimensions).

Plan 0: geometric models

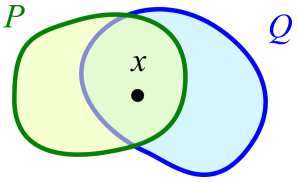
一阶逻辑语法 $\mathscr{L}$ 不是必需的，只需某种将 model **改写** 的能力 (27)

可以有很多不同的变种，例如：

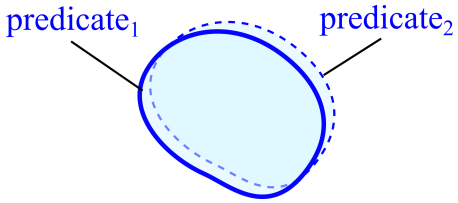
$$\begin{array}{ccc} \text{逻辑语法 } \mathscr{L} & & \text{graph re-writer} \\ \downarrow & \approx & \downarrow \\ \text{partial model } \mathscr{M} & & \text{graph model} \end{array} \tag{28}$$

[†]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.

Consider a kind of what I call **geometric models**, as typically exemplified by Venn diagrams:

$$P(x) \wedge Q(x) \quad \cong \quad \text{  } \tag{29}$$

The motivation for introducing geometric models: Since we need to perform search among logic formulas, it may be desirable to represent formulas such that they can continuously "morph" into each other:



(30)

In classical logic this is not possible. For example, $\text{Love}(x, y)$ and $\text{Like}(x, y)$ may be similar in meaning, but the transition from one to the other requires a discrete "jump".

We propose to use **points** to represent first-order objects (*eg* "John") and **regions** (*ie*, point sets, open sets) to represent predicates and relations (*eg* "Love"). It is possible to use a *dual* model in which points and regions are swapped.

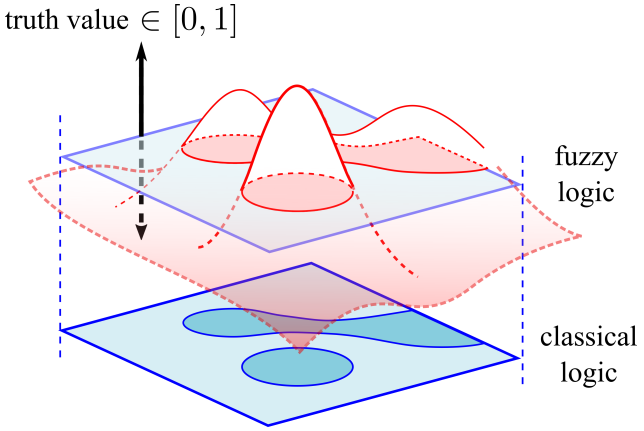
A caveat is that the points and regions must be *labelled* entities. For example, it is not sufficient to know that the point (John, Mary) belongs to a relation, but it matters whether that relation is Love or Hate. In other words, the *identity* of the region matters.

Previously I made the mistake of forgetting the labels associated with geometric regions, thus falsely simplifying the algorithm and claimed success. It turns out that the presence of labels forces us to handle the rewriting function on the **syntactic** level, and the geometric shapes of models do not seem to make an essential difference (to my disappointment). So this idea needs re-consideration.

总括来说，这个方案是：

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

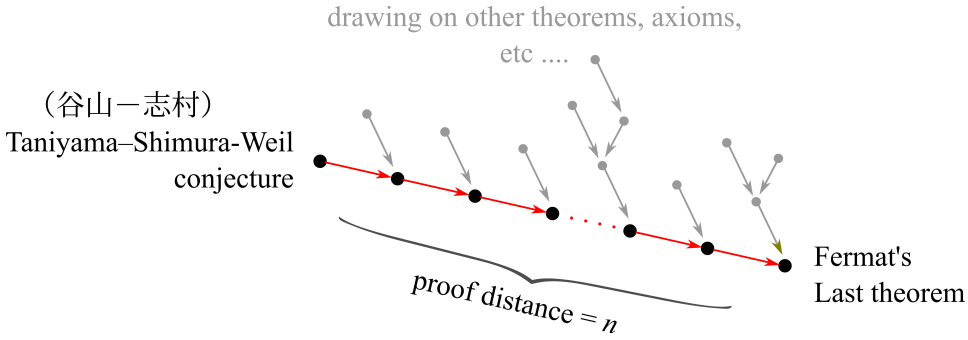
但我们需要一个 可以连续变化的 model of set theory，或许可以用某种 fuzzy topology 来达成？例如：



(32)

其需要定义的 boundary of regions 维数 高了 1 度。

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（它还取决於 KB 内的其他式子），但根据 Turing 在 1936 年证明的的 halting problem 定理，这个 distance 是**不可计算的**。‡ 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 $\vec{u} = \Theta$ 类似於逻辑式子，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？这取决於 fuzzy geometric model 的 rewriting function space Θ 是不是 **metrizable** (*cf* [Goubault-Larrecq 2013]). 在未来我会更详细研究这点。



(33)

The "plateau problem": From [Bergadano and Gunetti 1996]: Researchers have discovered that, during the inductive learning of logic formulas, it may happen that in the body of the correct clause, the gain remains low and constant (on a plateau) until one final important literal concludes the computation and brings the gain to a local maximum. An example is the following target clause:

```
append(X,Y,Z) :- list(X), head(X,X1), tail(X,X2),
append(X2,Y,W), cons(X1,W,Z).
```

(34)

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 (35)

$$\begin{aligned} \text{model } \mathscr{M} &= \text{符号逻辑式子} \\ \text{rewriter} &= \text{符号逻辑式子} + \text{经典 AI 逻辑引擎} \end{aligned} \tag{36}$$

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.

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。

‡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].

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”）：

$$\forall x, y.\ x \heartsuit y \wedge \neg y \heartsuit x \rightarrow \frownie x \tag{37}$$

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**：

$$a \heartsuit b \wedge \neg b \heartsuit a \tag{38}$$

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

$$a \heartsuit b \wedge \neg b \heartsuit a \rightarrow \frownie a \tag{39}$$

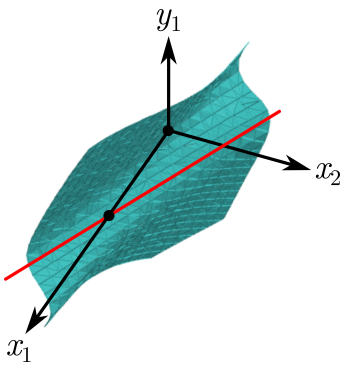
这两个动作（**compare** 和 **copy**）都是用神经网络很难做到的。但它们是 variable substitution 的本质，也是谓词逻辑 麻烦之处。换句话说，很难用一个 monolithic 的 end-to-end 神经网络 一口气完成这两个动作：

$$a \heartsuit b \wedge \neg b \heartsuit a \longrightarrow \frownie a \tag{40}$$

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：

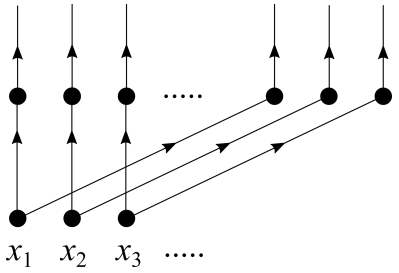
$$\vec{F}:\ (x_1, ..., x_i, ..., x_n) \mapsto (y_1, ..., y_j, ..., y_n) \tag{41}$$

这要求神经网络的函数曲面穿过某些 diagonal 线，如下图：



(42)

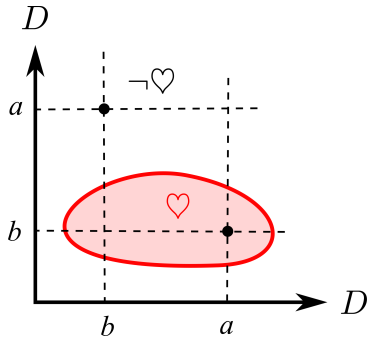
以下是一个简单的 copier 神经网络（所有权重 $= 1$，其他权重 $= 0$ 没有显示）：



(43)

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 §：



$$\Rightarrow \ominus a \tag{44}$$

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

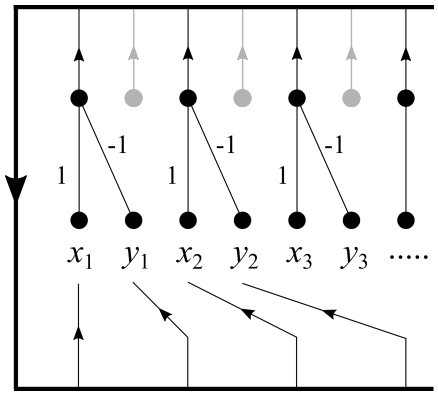
$$\heartsuit(\vec{x}) = \begin{cases} 0 & \vec{x} \notin \text{region} \\ 1 & \vec{x} \in \text{region} \end{cases} \tag{45}$$

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

$$\begin{array}{rcl} \vec{p}_1 = (\vec{a}, \vec{b}) & \in & \heartsuit \\ \Big\Vert & & \\ \vec{p}_2 = (\vec{b}, \vec{a}) & \in & \neg\heartsuit \end{array} \tag{46}$$

§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....

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）：



(47)

在 iterate n 次之后，最左边的输出 会是 $\vec{x} \overset{?}{=} \vec{y}$ 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。

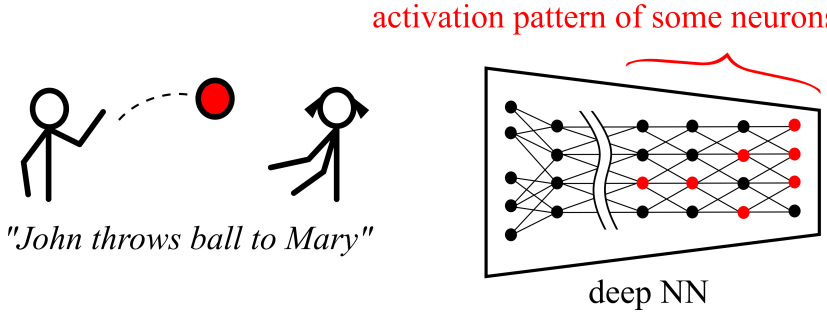
「分布式」知识表述

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：

$$\vec{x} = (x_1, x_2, ..., x_{10}) \tag{48}$$

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**：



(49)

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：

$$\text{A 掷球给 B} \Longleftrightarrow \begin{cases} \text{A 手臂挥动} \wedge \\ \text{球离开 A 的手} \wedge \\ \text{球在半空飞} \wedge \\ \ldots \end{cases} \tag{50}$$

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：

$$\boxed{\text{"neat" proposition}} \quad p \Longleftrightarrow \begin{cases} q_1 \wedge \\ q_2 \wedge \\ \ldots \wedge \\ q_n \end{cases} \quad \boxed{\text{distributed propositions}} \tag{51}$$

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：

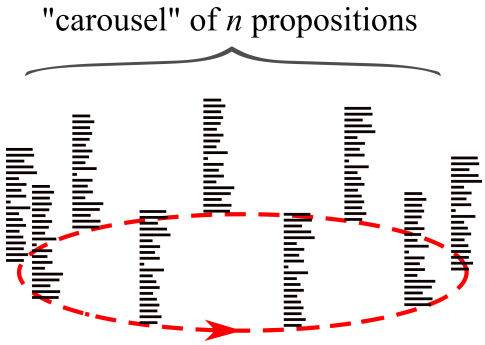
(52)

「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）：



(53)

总之，纯粹用 NN 模拟 STM，明显地很麻烦。

Graph NNs

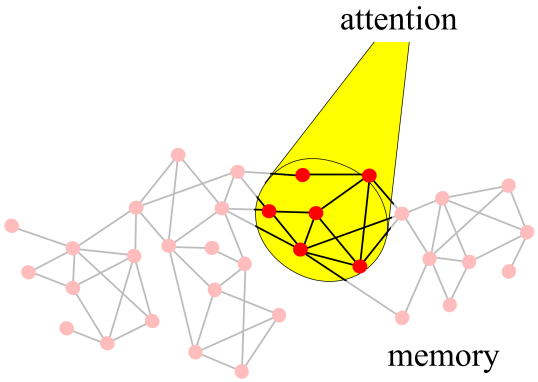
用 graph 做记忆体（包括短期和长期记忆） (54)

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。

$$\begin{aligned} \text{model } \mathscr{M} &= \text{graph} \\ \text{rewriter} &= \text{deep NN} = \text{graph neural network} \end{aligned} \tag{55}$$

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：



(56)

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：

$$\boxed{\text{graph}} \quad \xrightarrow{\text{embed}} \quad (x_1, ..., x_m) \quad \boxed{\text{vector}} \tag{57}$$

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）。

Quiver representations

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：

$$\text{John} \underset{\neg\heartsuit}{\overset{\heartsuit}{\rightleftarrows}} \text{Mary} \quad \Longrightarrow \quad V_1 \underset{M_2}{\overset{M_1}{\rightleftarrows}} V_2 \tag{58}$$

其中 V_1, V_2 是向量空间，$M_1, M_2 \in GL(\mathbb{R})$ 是 矩阵。

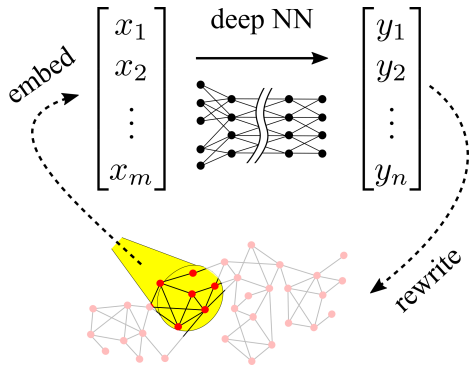
The vector spaces V_i can be stacked up vertically to form a single big vector space V. Then all the relations fit into an endomorphism $V \to V$.

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。

¶线性是指符号形式上，例如 tree 可以表示成线性的一行

整体运作是这样的：



(59)

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**。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）。

馀下 2 节是一些 数学 背景知识....

What is topos theory?

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]）

An **elementary topos** ("elementary" as in "elements" of a set) is a **Cartesian-closed category** (CCC) with a **sub-object classifier** Ω.

A Cartesian-closed category is, roughly speaking, where one can form arbitrary:

- **Cartesian products** $A \times B$, and
- **exponentiation** B^A

[‖]Elementary 是集合中「元素」的意思

where $B^A \simeq A \to B$ is the class of all functions from A to B.

The sub-object classifier Ω allows a topos to handle **subsets** as in set theory. In the category **Set**, Ω is $\{\top, \bot\}$, representing True and False.

For example, "Love" is a **relation** within $D \times D$, where D is the universe set of all "people". One can think of $D \times D$ as the "full" relation, then Love $\subset D \times D$ is its **subset**. Thus sub-objects enable elementary toposes to model relation algebra and first-order logic. A standard textbook on topos theory is [Goldblatt 1984, 2006].

Some history

「*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.

"In a sense logic is a special case of geometry." — Bill Lawvere.

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 $\mathscr{A}$ 上的 pre-sheaf 可以定义为一个 **functor**:

$$\widehat{\mathscr{C}} : \mathscr{C}^{\text{op}} \to \mathbf{Set} \tag{60}$$

亦即是 范畴 $\mathscr{C}$ 上的一「层」set-valued functions. 它是 "functor" 所以处理了那些 collating. 这个 functor 是 contravarient 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。

Sub-object classifier

This commutative diagram defines the sub-object classifier Ω:

$$\begin{array}{ccc} A & \xrightarrow{\ f\ } & D \\ {\scriptstyle !}\downarrow & \lrcorner & \downarrow{\scriptstyle \chi_f} \\ 1 & \xrightarrow{\ true\ } & \Omega \end{array} \tag{61}$$

**around 1960-70

Pulling *true* back along χ_f yields the set $\{x : \chi_A(x) = 1\}$ which is just A.

The arrow χ is called the **classifying** arrow of the sub-object A; It can be thought of as taking exactly the part of D that is A to the "point" t of Ω.

We form a collection, the sub-objects of D:

$$\mathrm{Sub}_{\mathscr{C}}(D) = \{[f] : f \text{ is a monic with } \mathrm{cod} f = D\} \tag{62}$$

If $\mathscr{C}$ has pullbacks (of monomorphisms along arbitrary morphisms in $\mathscr{C}$) then $\mathrm{Sub}_{\mathscr{C}}(D)$ extends to a functor $\mathrm{Sub}_{\mathscr{C}} : \mathscr{C}^{\mathrm{op}} \to \mathbf{Set}$ by putting $\mathrm{Sub}_{\mathscr{C}}(f)([m]_\sim) = [f^*m]_\sim$ where f^*m is the pullback of m along f. (??) [Streicher 2006] p.79.

The pullback condition is equivalent to requiring the contra-variant sub-object functor,

$$\mathrm{Sub}_{\mathscr{C}}(-) : \mathscr{C}^{\mathrm{op}} \to \mathbf{Sets} \tag{63}$$

(which acts by pullback) to be **representable**, ie:

$$\mathrm{Sub}_{\mathscr{C}}(-) \cong \mathrm{Hom}_{\mathscr{C}}(-, \Omega). \tag{64}$$

The required isomorphism is just the pullback condition stated in the definition of a sub-object classifier [Awodey 2006] p.175.

A contra-variant (or co-variant) presheaf $F : \mathscr{C}^{\mathrm{op}} \to \mathbf{Set}$ is called **representable** if it is isomorphic to $\mathrm{Yon}_{\mathscr{C}}(I) = \mathscr{C}(-, I)$ (or $\mathscr{C}(I, -)$) for some $I \in Obj\ \mathscr{C}$.

Example. If $\mathscr{C}$ is a monoid M (considered as a category) then Ω consists of all right ideals in M, *ie*, subsets I of M such that $x \in I$ and $y \in M$ implies $xy \in I$, and $\Omega(x)(I) = \{y \in M | xy \in I\}$. [Streicher 2006] p.83. (??)

Example. If $\mathscr{C}$ is a group then according to the above example, the truth value object Ω of the topos $\widehat{\mathscr{C}}$ of G-actions has $\{G, \varnothing\}$ as underlying set because G and $\varnothing$ are the only right ideals in G which, moreover, are left invariant by all actions of group elements.

Logic in a topos

In (higher-order) predicate logic we are concerned with sequents of the form:

$$\Gamma \rhd \Phi \vdash \phi \tag{65}$$

where Φ is the **assumption**, ϕ the **conclusion**, and Γ is a **variable context** and specifices which free variables are allowed in Φ and ϕ:

$$\overbrace{x : \mathbb{N}, y : \mathbb{N}}^{\text{variable context}} \ \rhd \ \mathrm{odd}(x) \wedge \mathrm{odd}(y) \vdash \mathrm{even}(x + y) \tag{66}$$

For every **type** α in the signiture we have an object $[\![\alpha]\!]$ of $\mathscr{C}$.

For every **proved term** $\Gamma \vdash M : \alpha$, we have a morphism in $\mathscr{C}$:

$$[\![\Gamma \vdash M : \alpha]\!] : [\![\Gamma]\!] \to [\![\alpha]\!] \tag{67}$$

For $A \in \mathrm{Obj}\ \mathscr{C}$, the set of **predicates** in $A = \mathrm{Sub}_{\mathscr{C}}(A) \cong \mathscr{C}(A, \Omega)$, with a partial order $\leq_A$ on it.

Fibration

[Jacobs 1999]

Predicates on sets can be organized in a category called **Pred**:

- **Objects** are pairs (I, X) where $X \subseteq I$ is a subset of a set I; in this situation we consider X as a predicate on a type I, and write $X(i)$ for $i \in X$ to emphasize that an element $i \in I$ may be understood as a free variable in X. When I is clear from the context, we sometimes write X for the object $(X \subseteq I)$.
- **Morphisms** $(I, X) \to (J, Y)$ are functions $u : I \to J$ between the underlying sets satisfying:

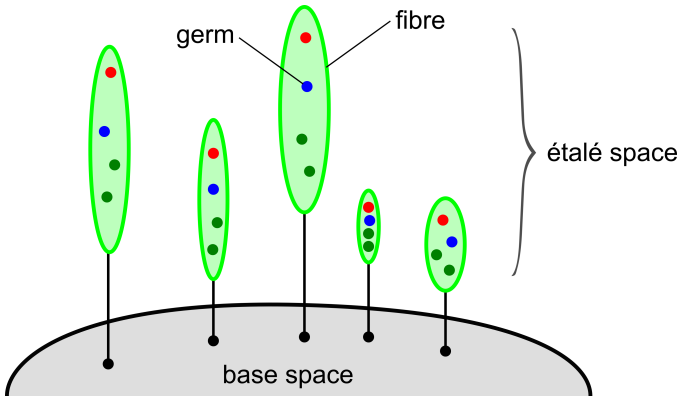
$$X(i) \text{ implies } Y(u(i)), \text{ for each } i \in I. \tag{68}$$

  Diagrammatically, this condition on such a function $u : I \to J$ amounts to the existence of a necessarily unique (dashed) map:

$$\begin{array}{ccc} X & \longrightarrow & Y \\ \downarrow & & \downarrow \\ I & \xrightarrow{u} & J \end{array} \tag{69}$$

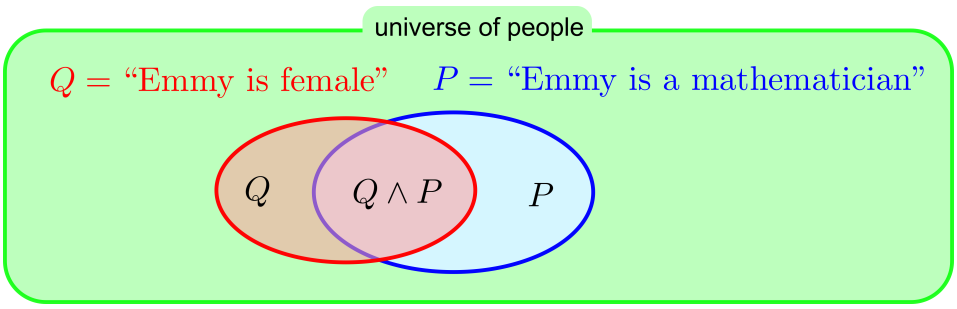
  indicating that u restricts appropriately.

Illustration of a **fibration**:



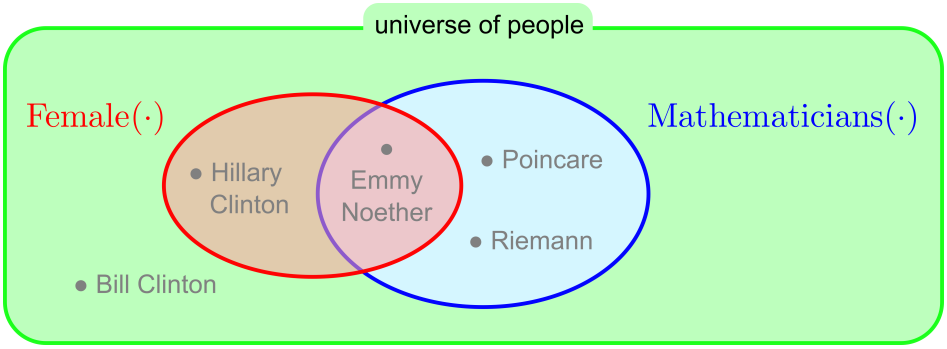
(70)

Propositional logic is a kind of **topological** (open set) structure:



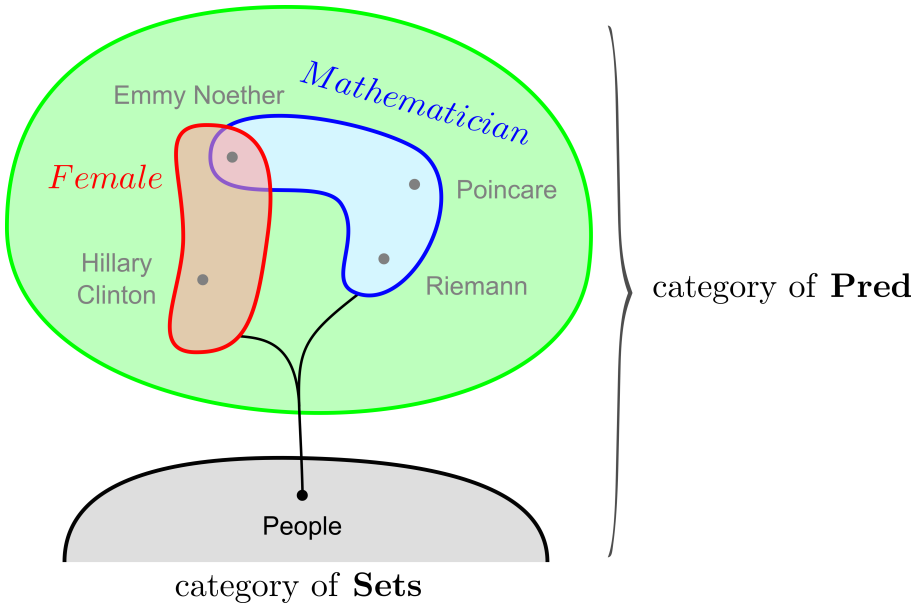
(71)

Inside a proposition, there is the **predicate** structure:



(72)

Predicate logic could be regarded as a **fibration** over propositional structure, denoted as $\begin{smallmatrix}\textbf{Pred}\\ \downarrow \\ \textbf{Set}\end{smallmatrix}$:



(73)

Quantifications are adjoints

A pair of monotone functions $f : X \to Y$ and $g : Y \to X$ between pre-ordered sets determines an **adjunction**, if $\forall x \in X, y \in Y$:

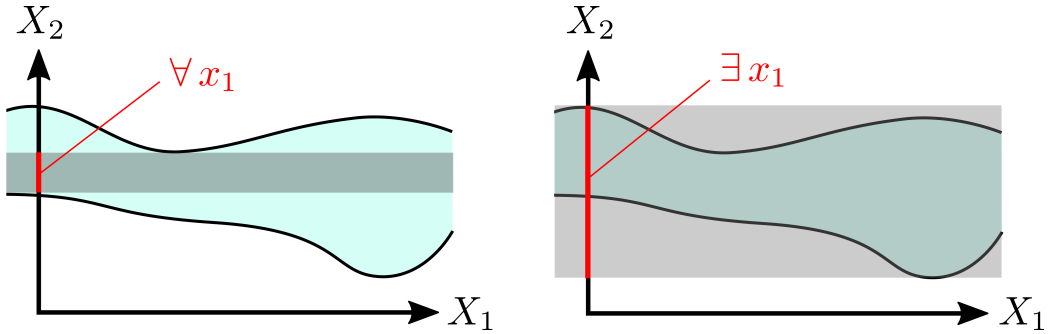
$$f(x) \leq y \text{ in } Y \quad \Leftrightarrow \quad x \leq g(y) \text{ in } X \tag{74}$$

in which case we write $f \dashv g$ and say "f is left-adjoint to g", or "g is right-adjoint to f". (Mnemonic: the *left* adjoint appears on the *left* of $\leq$ and *vice versa*)

Lawvere discovered the following adjunctions:

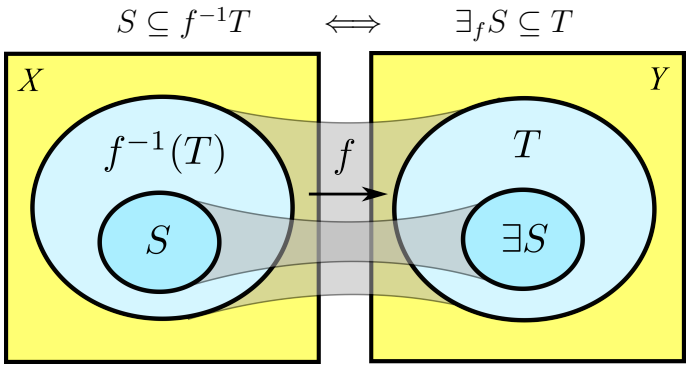
$$\exists \dashv * \dashv \forall \tag{75}$$

In cylindric algebra, the quantifiers $\forall$ and $\exists$ can be interpreted as **projections** to a component (X_2) of the domain:

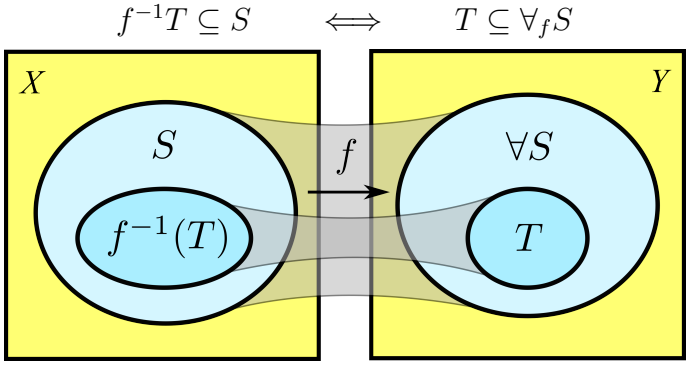


(76)

The following 2 formulas are from [Abramsky and Tzevelekos 2011]. For all $S \subseteq X, T \subseteq Y$:

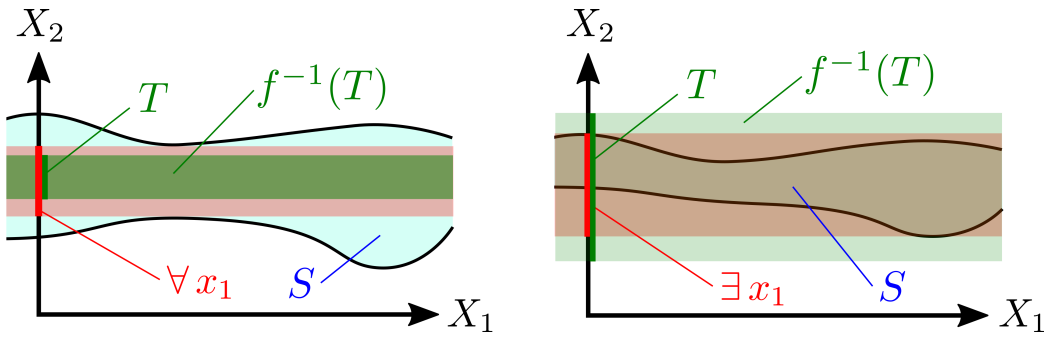


(77)



(78)

They can be interpreted in a "cylindrical" way: Let $X = X_1 \times X_2$ and $Y = X_2$. Then f is a projection $X_1 \times X_2 \to X_2$:



(79)

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 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
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

(80)

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：

$$\boxed{\wedge \text{ rule}} \quad \frac{\Phi \vdash \Psi,\ \Phi \vdash \chi}{\Phi \vdash (\Psi \wedge \chi)} \tag{81}$$

$$\boxed{\exists \text{ double rule}} \quad \frac{\Phi \vdash_{\vec{x},y} \Psi}{\exists y\, \Phi \vdash_{\vec{x}} \Psi} \tag{82}$$

$$\boxed{\text{Frobenius axiom}} \quad \Phi \wedge \exists y\, \Psi \vdash_{\vec{x}} \exists y\, (\Phi \wedge \Psi) \tag{83}$$

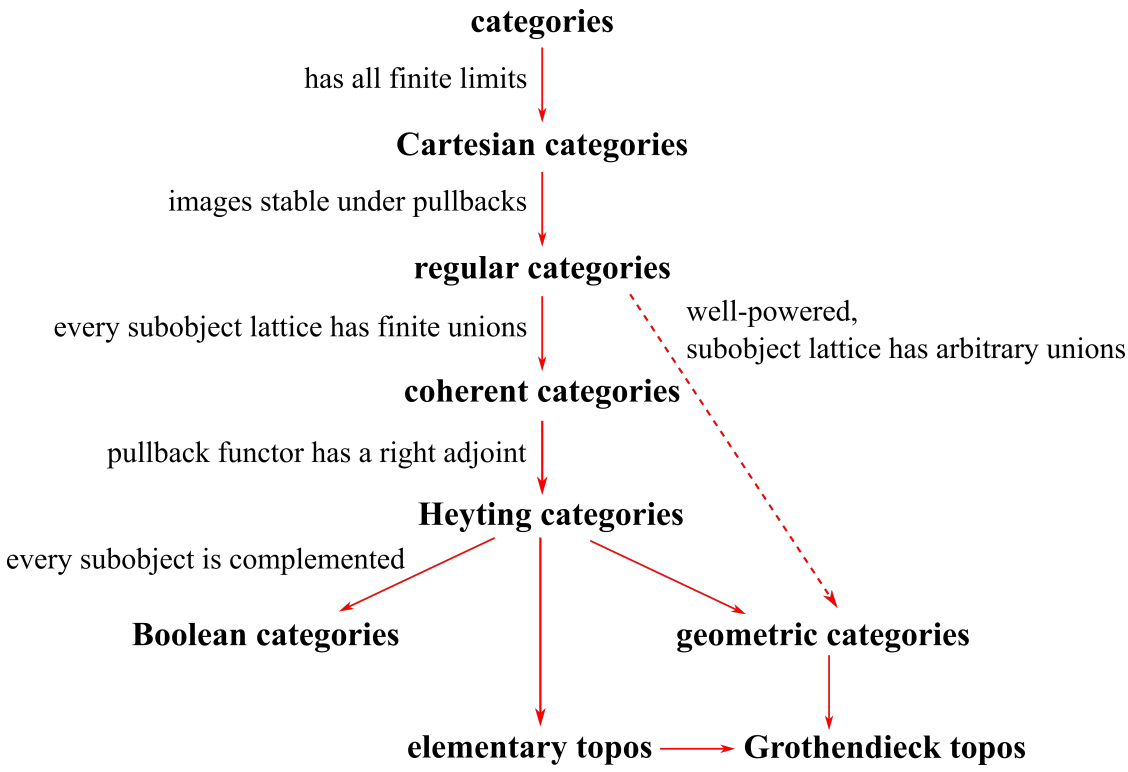
Tarski 的模型论 将 first-order syntax 「对应」到 集合论 的 **结构** (structures) 上。由此推广，不同的 逻辑 syntax 对应於不同的 结构范畴：

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

(84)

"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。

以下是根据 [Caramello 2018] Ch.1 整理出来的一张关系图：



(85)

Grothendieck's idea of topos is a *stronger* notion than elementary topos.

Fuzzy logic

📖 From §5.2.4 of [Bělohlávek, Dauben, and Klir 2017].

In any topos one may define arrows which play the role of **truth functions** of logical connectives on Ω. In this sense, each topos "internalizes" a logic. It is a striking fact that, in general, Ω becomes a Heyting algebra and thus the internal logic of topoi is intuitionistic logic.

An early attempt of a category of fuzzy sets is $\mathbf{S}(L)$ proposed by Goguen. For a complete residuated lattice L, $\mathbf{S}(L)$ has as **objects** the pairs $\langle U, A\rangle$ where U is a set and A an L-set of U, *ie*, $A : U \to L$, *ie*, a subset of U fuzzified by L; and as **morphisms** $R : \langle U, A\rangle \to \langle V, B\rangle$ the L-relations $R : U \times V \to L$ satisfying $\bigvee_{u\in U} A(u) \otimes R(u, v) \leq B(v)$, along with $\circ$-composition. This is later shown to be a **quasi-topos**.

In 1981 Eytan described a category of **Heyting-algebra-valued sets**, similar to Goguen's $\mathbf{S}(L)$, and claimed that it forms a topos, which was disproved by Pitts the next year.

Graphs

The category of graphs is a functor category:

$$\mathbf{Graphs} = \mathbf{Sets}^{\Gamma} \tag{86}$$

where Γ is a category pictured as follows:

$$\bullet \rightrightarrows \bullet \tag{87}$$

It has exactly 2 objects and 2 distinct arrows. It follows from this that **Graphs** is Cartesian-closed [Awodey 2006] p.143.

Homotopy type theory

[Rodin 2014]

2 continuous maps:

$$A \underset{g}{\overset{f}{\rightrightarrows}} B \tag{88}$$

between 2 spaces A, B from **Top** are called **homotopical** when there exists a homotopy between them. A homotopy h between 2 continuous maps $f, g : A \to B$ in **Top** is a continuous map $h : A \times [0, 1] \to B$ such that $h(0) = f$ and $h(1) = g$.

By identifying all homotopic maps in **Top** one gets the **homotopy category hTop**.

Quillen in 1967 axiomatized homotopy theory as the **model category**. Awodey and Warren showed that every model category admits an internal language, which is a form of Martin-Löf type theory.

Voevodsky provided this inductive definition:

1. A space A is called **contractible** (a.k.a. a space of h-level 0) when there is a point $x : A$ connected by a path with each point $y : A$ in such a way that all these paths are homotopic.
2. We say that A is a space of h-level $n + 1$ if for all points x, y the path spaces $\mathrm{Path}_A(x, y)$ are of h-level n.

Then we have these hierarchical levels:

- **Level 0:** up to homotopy equivalence there is just one contractible space that we call "point" and denote pt
- **Level 1:** up to homotopy equivalence there are 2 spaces at this level: the **empty space** $\varnothing$ and the **point** pt. We call them **truth values**. We also refer to types of this level as **properties** and **propositions**. Propositional logic lives at h-level 1
- **Level 2:** Types of this level are characterized by the property that their path spaces are with empty or contractible. So such types are disjoint unions fo contractible components (points), or in other words **sets** of points. This will be our working notion of set available in this framework.
- **Level 3:** Types of this level are characterized by the property that their path spaces are sets (up to homotopy equivalence). These are obviously (ordinary flat) **groupoids** (with path spaces hom-sets)
- **Level 4:** Here we get 2-groupoids

  $\vdots$
- **Level n + 2:** n-groupoids

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 的解决办法是给 domain D endow with a **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。

Scott worked on Boolean-valued models of set theory and the idea of replacing Boolean by **Heyting algebras**. His considerations lead to topoi structures and the concept of **Heyting-valued sets** which play an important role in the development of fuzzy topos theory.

From [**Pitts1991**] §4-5:

There is a correspondence between logic theories and pre-ordered sets, tracing back to the correspondence between classical propositional logic and Boolean algebras.

Lawvere's dictum: *"categories are theories and models are functors"*

..... 【未完待续】 

Conclusion and further questions

- We saw that the model structures are "grounded" in the sense that they don't contain variables. Does this restriction result in a sub-category of toposes? What is the categorical characterization of such structures?
- Does Lawvere quantification generalize to cases where the domain may not be Cartesian products or where f is not a projection function? This may allow for more variations of geometric models.
- Fuzzy logic and its relation to topos theory.

References

Abramsky and Tzevelekos (2011). Introduction to categories and categorical logic. In: *New structures for physics*. Ed. by Coecke. Chap. 1.

Awodey (2006). Category theory.

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

Bělohlávek, Dauben, and Klir (2017). Fuzzy logic and mathematics – a historical perspective.

Bergadano and Gunetti (1996). Inductive logic programming - from machine learning to software engineering. MIT.

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 (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.
Jacobs, Bart (1999). Categorical logic and type theory. Elsevier.
Lawvere and Rosebrugh (2003). Sets for mathematics. Cambridge.
Lawvere and Schanuel (1997, 2009). 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.
Rodin, Andrei (2014). Axiomatic method and category theory.
Streicher (2006). Domain-theoretical foundations of functional programming. World Scientific.
Vickers (1989). Topology via logic.