《AGI 逻辑导论》

YKY

May 23, 2022

Summary

- 用比较简单的方法讲解 范畴论下逻辑的结构，这亦叫 categorical logic. 这是目前数学上描述逻辑结构的最满意的形式。了解逻辑的抽象结构之后，可以将它应用到 AGI 或神经网络的设计上。

Contents

0 Background

我们想 **训练** 一个智能系统，训练 是一个 **机器学习** 的过程，也是一个 **optimization** problem, 目标是将 **长期的奖励总和** 最大化：

$$\text{maximize:} \int_0^\infty R\,dt \tag{0.0.1}$$

where $R(t)$ = reward at time t. $\int_0^\infty$ 表示 计算 累积奖励的 **time horizon**. （我使用了微分的形式，实际应用通常是离散形式，但两者基本一样，不必深究）

俗语说「棋屎贪吃卒」，在开局初期吃卒，可能导致 N 步之后被将死，这是**愚蠢**的行为。所以 (0.0.1) 式 令 系统必需顾及长远的利益，遂迫使它学习 **智慧**。

Architecturally, the AI is a **dynamical system** that constantly updates its "state" x via: [*]

$$\dot{x} = f(x) \tag{0.0.2}$$

或者用离散形式表示：

$$x_{t+1} = F(x_t) \tag{0.0.3}$$

F 叫作 transition function. 或者更形象地表示：

$$\begin{tikzcd} x \arrow[loop above, "F"] \end{tikzcd} \tag{0.0.4}$$

Our goal is to **learn** the function F, implemented as a **deep neural network**. F 包含智能系统内的所有**知识**。

[*] Part of the state x contains **sensory input** and **action output** that allow the AI to interact with the external environment.

1 Structure of logic

My thesis is that the state x of the AI system is consisted of **logic propositions** and that F plays the role of the **logic consequence** operator $\vdash$:

$$\boxed{\text{propositions}} \overset{F}{\vdash\!\!\longrightarrow} \boxed{\text{propositions}} \tag{1.0.1}$$

So our goal now is to elucidate the structure of $\vdash$. Currently the most elegant formulation is given by **categorical logic** or **topos theory**.

在接下来的篇幅，我会勾划一个 对於 AGI 来说是完整的 逻辑理论，而这理论 的中心思想 就是 Curry-Howard isomorphism....

2 Curry-Howard correspondence

The Curry-Howard isomorphism expresses a **connection** between logic and functions, specifically functions that map proofs to more proofs, thus implementing inference as a **process**. The correspondence <u>cannot be proven mathematically</u>; Its status in mathematics is akin to the postulates of **quantum mechanics**: they relate certain mathematical structures, such as Hilbert space vectors and Hermitian operators, to **physical reality**. Without these postulates, the Schrödinger equation is just a differential equation. Analogously, the Curry-Howard isomorphism connects mathematical structures to **logic**:

$$\begin{array}{ccc} \boxed{\text{mathematics}} & \xrightarrow{\text{quantum theory}} & \boxed{\text{physical reality}} \\ \boxed{\text{mathematics}} & \xrightarrow{\text{Curry-Howard theory}} & \boxed{\text{logic}} \end{array} \tag{2.0.1}$$

Here logic is viewed as some kind of *empirical* phenomenon, whose rules are not known to us *a priori*, but are distilled from our experience.

简单来说：当我们做 逻辑思考时，表面上有一种语法上 (syntax) 的形式，即 $A \Rightarrow B$:

$$\begin{array}{cc} \boxed{\text{logic}} & A \Longrightarrow B \\ & \text{- - - - - - - - - - - - -} \\ \boxed{\text{program}} & \blacksquare \overset{f}{\longmapsto} \blacksquare \end{array} \tag{2.0.2}$$

而在这 语法「底下」，还有一个 **运算**，它可以看成是执行 **证明** (proof) 的工作，它将 A 的证明 map 到 B 的证明。*

一个传统的数学函数，例如 $f(x) = x + 2$ 用我们惯常的符号表示为：

$$\frac{f : \mathbb{R} \longrightarrow \mathbb{R}}{x \longmapsto x + 2} \tag{2.0.3}$$

这不是新的。类似地，一个逻辑式子：

$$x \text{ 是偶数} \Longrightarrow x + 2 \text{ 是偶数} \tag{2.0.4}$$

也不是新的。但如果将「x 是偶数」这个**命题**，看成是一个**类型**或**集合**，里面有个**证明** (witness)，这个看法是新的：

$$\frac{\boxed{x \text{ 是偶数}}}{\blacksquare} \tag{2.0.5}$$

This is called the **Brouwer-Heyting-Kolmogorov (BHK) interpretation**. 由於这个想法比较 subtle，它不断被重复发现很多次，命名者可以包括：Brouwer-Heyting-Kolmogorov-Schönfinkel-Curry-Meredith-Kleene-Feys-Gödel-Läuchli-Kreisel-Tait-Lawvere-Howard-de Bruijn-Scott-Martin-Löf-Girard-Reynolds-Stenlund-Constable-Coquand-Huet-Lambek ....

Once we associate a mathematical function to the logic symbol $\Rightarrow$, such a correspondence induces a lot of other correspondences: propositions are abstract spaces, and as spaces they may possess topological structure, ... and so on. Thus the Curry-Howard isomorphism induces an entire enterprise of applying mathematics to the study of logic.

根据 HoTT (homotopy type theory)，一个命题 可以**有**或**没有**证明；如果有，则它的证明都是一样的，所以 经典逻辑命题 只可以取值 **真**或**假**。我的理论推断：模糊逻辑的取值 $\in [0, 1]$ 是因为 fuzzy 命题 可以有「部分」的证明（见 §7）。

* 为了避免符号累赘，我将这些 “proof witness” 都记作 $\blacksquare$，但它们每个是不同的。

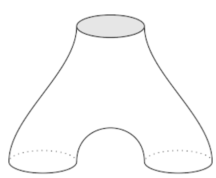
John Baez (1961-)

从另一角度看，Curry-Howard isomorphism 可以看成是 某些 **状态**（states，例如 A）和状态之间的 **转换** （transitions，例如 $A \xrightarrow{f} B$）之间的 **对偶**。而这种对偶 不断在 截然不同的范畴里出现：

logic	**computation**	**category theory**	**physics**	**topology**
proposition	type	object	system	manifold
proof	term	morphism	process	cobordism

(2.0.6)

前两个就是 Curry-Howard，第三个 是 Lambek 加上去的，其馀的来自 John Baez & M. Stay 的论文：*Physics, Topology, Logic and Computation: a Rosetta stone* [2010]. 例如在 physics 里面是 Hilbert space 和 operators 的对偶；在 topology 里面，cobordism 的著名例子就是这个 “pair of pants”：



(2.0.7)

In string theory，它表示上面的 strings 变成下面的 string 的「时间过程」。

2.1 Type theory

描述 **program** 或 **computation** 的语言叫 type theory. 例如在一般的 编程语言 里可以有这样的一句：

```
define length(s: String): Integer = { .... }
```

(2.1.1)

意思是说 length() 是一个函数，输入 String，输出 Integer.

在数学里 我们描述 **函数** 时会用：

$$f : A \to B \tag{2.1.2}$$

这个表达式其实就是 type theory 的一般形式：

$$\overbrace{t}^{\text{term}} : \overbrace{T}^{\text{type}} \tag{2.1.3}$$

而这个 notation $t : T$ 其实也可以写成 $t \in T$（但不正统而已）。

换句话说，types 就是 **集合**，terms 是集合中的 **元素**。

更一般地，一个 type theory 的句子 可以包含 type **context**：

$$\overbrace{x : A}^{\text{context}} \vdash \overbrace{f(x) : B}^{\text{type assignment}} \tag{2.1.4}$$

意思就像在 program 的开头 "declare" 一些 变量 的类型，然后 program 就可以被 **赋予** 后面的 类型。

这个 $\vdash$ 的过程 称为 **type assignment**，而这就是 type theory 做的全部工作。

λ-calculus

在一个 program 里，除了定义 类型，还需要定义 **函数**。这件工作是由 λ-calculus 负责。

λ-calculus 可以定义函数 而不需要提及它的「名字」。例如，用数学式表达：

$$f(x) \triangleq x^2 \tag{2.1.5}$$

它的 λ-表达式就是：

$$f \triangleq \lambda x.\, x^2 \tag{2.1.6}$$

注意：在 λ-表达式里，不需要提到 f 的「名字」。

λ-calculus 是由 Alonso Church 发明，目的是研究数学上 **substitution** 的性质。Substitute 是每个中学生都懂得做的事，但要用数学表达出来却是出奇地麻烦。

同时，Church 发现 λ-calculus 是一种「万有」的计算形式，和 **Turing machines** 等效。「AI 之父」John McCarthy 用 λ-calculus 发展出 **Lisp** 语言，它是所有 functional programming language 的鼻祖。

Curry-Howard correspondence

在 Curry-Howard 对应下，type A 就是 逻辑命题 A，type A 或 集合 A 里面的 元素 是其 **证明** (proof, or proof witness)。

而，$A \Rightarrow B$ 也是 逻辑命题，它对应於 the function type $A \rightarrow B$，也可以写作 B^A，而这个 type 或 集合 里面的 元素 就是一些 函数 $f : A \rightarrow B$. 如果 有一个这样的函数存在，则 type $A \rightarrow B$ 有「住客」(inhabited)，换句话说 $A \Rightarrow B$ 有 **证明**。

2.2 Intuitionistic logic

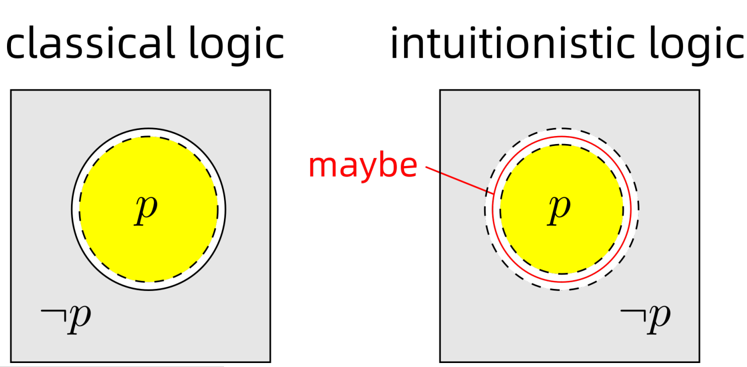
Curry-Howard isomorphism 揭示了 type theory 和 **intuitionistic logic** （直觉主义逻辑）之间的关系。这种逻辑的特点是没有 **排中律** (law of excluded middle, LEM)，或者等价地，**double negation**，即 $\neg\neg p \Rightarrow p$.

排中律 是说：$p \vee \neg p$ 是 恒真命题。但在 直觉主义 逻辑中，$p \vee \neg p$ 表示 p 的证明 **或** $\neg p$ 的证明，但有时候这两者都不知道（例如 现时仍未找到证明，或者不可能找到证明）。

附带一提：人们惊讶地发现，在直觉主义逻辑下，axiom of choice $\Rightarrow$ law of excluded middle. 换句话说，axiom of choice 和 直觉主义 也有内在的矛盾。

Topological interpretation

在 拓樸学 里，一般用 **open sets** 表示空间中的子集（这习惯起源自 Hausdorff 时期）。但 p 的 **补集** $\overline{p}$ 并不 open，所以要将 $\neg p$ 定义为 p 的补集的 **interior**，即 $\neg p \triangleq \overline{p}^{\circ}$. 於是 $p \cup \neg p \neq$ Universe: *



(2.2.1)

* diagram from the book: *Classical and Non-classical Logics – an introduction to the mathematics of propositions* [Eric Schechter 2005], p.126.

2.3 Higher-order logic

Propositional logic 的意思是：只有命题，但忽略任何 **命题内部** 的结构。

假设 p, q 是命题，命题逻辑的基本运算 就是 $p \wedge q, p \vee q, p \Rightarrow q, \neg p$.

First-order logic 的意思是：容许 这样的方法 构成 命题：

$$\overbrace{\text{IsHuman}}^{\text{predicate}}(\overbrace{\text{John}}^{\text{object}}). \tag{2.3.1}$$

Predicate 的意思是 **谓词**；谓词 是一些「有洞的命题」，它们被填入 objects 之后就变成完整的命题。类似地可以有 **多元**的 predicates，例如：

$$\text{Loves}(\text{John}, \text{Mary}). \tag{2.3.2}$$

First-order 指的是：$\forall, \exists$ 这些 **量词** 可以 **作用** 在 (first-class) objects 的类别上，例如（Mary 人见人爱）:

$$\forall x.\ \text{Loves}(x, \text{Mary}) \tag{2.3.3}$$

但 first-order logic 不容许 量词 作用在 predicates 的类别上，除非用 second-order logic.

一个 二阶逻辑的例子是「拿破仑 具有一个好将军应该具备的所有特质」:

$$\forall p.\ p(\text{Good General}) \Rightarrow p(\text{Napoleon}). \tag{2.3.4}$$

注意 p 是在 predicates 的类别之上量化的。

2.4 旧式 logic with type theory

Type theory 的历史还可以追溯更早。它起源於 Russell 为了解决 **逻辑悖论**，例如:「一个只帮自己不理发的人理发的理发师帮不帮自己理发?」这些 逻辑悖论 根源是在於：定义一样东西的时候，中途 **指涉** 了这个东西本身。这种不良的定义称作 **impredicative**. 为了避免不良定义，每个东西出现之前必需「宣告」它的类型，这就是 type theory 原来的目的。

在 Curry-Howard isomorphism 未被重视之前，有一种更简单地 用 type theory 定义 逻辑的方法。在这种方法下，逻辑命题 $p, q, p \wedge q$ 等 **直接用** terms 定义，而不是像 Curry-Howard 那样，逻辑命题 $=$ types，证明 $=$ terms.

在这情况下 type theory 处理的是 (first- or higher-order) predicate logic 的方面。这是说，例如：

$$\text{IsHuman}(\text{John}) \tag{2.4.1}$$

里面 IsHuman 是一个 函数 term，它输入一个 物体，输出它是不是「人」的 真值 (truth value) $\in \Omega = \{\top, \bot\}$. 因此 IsHuman 是一个 类型为 $\text{Obj} \to \Omega$ 的 term.

这种做法没有容纳 Curry-Howard isomorphism 的馀地。如果要做到后者，需要的是 Martin-Löf type theory....

2.5 Martin-Löf type theory

根据 Curry-Howard, 下面的 $A {\Rightarrow} B$ 是一个 逻辑命题，因而是一个 type:

$$\underset{\Omega}{\overbrace{\text{Human (Socrates)}}^{A}} \Rightarrow \underset{\Omega}{\overbrace{\text{Mortal (Socrates)}}^{B}} \tag{2.5.1}$$

但另方面，Human() 和 Mortal() 这两个 predicates 也需要借助 type theory 来构成命题，它们也是 types. 红色 $\to$ 和 蓝色 $\Rightarrow$ 的两个层次 是完全不同的 两码子事，但因为 Curry-Howard 而被逼 挤在一起。这就使得 type theory 好像「一心不能二用」。

在 “simple” type theory 里面可以 构造：

- sum type $A + B$
- product type $A \times B$
- function type $A \to B$

分别对应於 直觉主义逻辑的 $\vee, \wedge, \Rightarrow$. 这些是在 **命题逻辑** 层面的，已经「用尽」了 type theory 的法宝。

但 Human(Socrates) 也是由 Human() 和 Socrates 构成的命题，这构成的方法是用一个 arrow $\to$，但已经没有 arrow 可用。

Martin-Löf 提出的解决方案是 引入新的 **type constructors**：

- **dependent** sum type Σ
- **dependent** product type Π

Dependent sum $\displaystyle\sum_A B$ 里面 B 的类型 depends on A. 整个 family of A 的 + 的结果变成类似 product $A \times B$.

Dependent product $\displaystyle\prod_A B$ 里面 B 的类型 depends on A. 整个 family of A 的 $\times$ 的结果变成类似 exponentiation B^A.

Dependent products can be used to define **predicates** such as Human() and Mortal(). They are of type $\text{Obj} \to \Omega = \Omega^{\text{Obj}} = \displaystyle\prod_{\text{Obj}} \Omega$. *

一个很漂亮的结果是：如果用 $\sum_A B$ 和 $\prod_A B$ 定义 逻辑命题，则这些 types 如果被 inhabited 的话，分别对应於 $\exists A.B(A)$ 和 $\forall A.B(A)$. 这是因为：如果 $A \times B$ inhabited，表示**至少存在**一个 $B(A)$；而如果 B^A inhabited，则存在一个函数，将**任意的** A send to B.

Per Martin-Löf (1942-) was the first logician to see the full importance of the connection between intuitionistic logic and type theory.

Per Martin-Löf (1942-) 

2.6 Arithmetic-logic correspondence

很多人都知道，经典逻辑中 $\wedge, \vee$ 对应於 **算术运算** $\times, +$（也可以看成是 fuzzy logic 的 $\min, \max$.）其实这就是 George Boole 尝试将 逻辑 变成 某种**代数** 的原因。

* Note that “objects” here mean logic objects, not objects in category theory.

较少人知道的是 $A \Rightarrow B$ 也对应於 B^A：[*]

$$
\begin{array}{|c|c|c|c|}
\hline
A & B & A \Rightarrow B & B^A \\
\hline
\hline
0 & 0 & 1 & 0^0 = 1 \\
\hline
0 & 1 & 1 & 1^0 = 1 \\
\hline
1 & 0 & 0 & 0^1 = 0 \\
\hline
1 & 1 & 1 & 1^1 = 1 \\
\hline
\end{array}
\tag{2.6.1}
$$

其中 0^0 是「不确定式」，但按照 组合学 惯例可以定义为 1.

这个惊奇的「巧合」似乎再一次证实 Curry-Howard correspondence 是正确的；特别地，它意味 $\Rightarrow$ 应该看成是 **函数**，即所谓 "functional interpretation of logical deduction."

2.7 The problem of material implication

更详细观察，table (2.6.1) 里面 A 和 B 的 truth values 可以看成是它们的 types 有没有 **inhabitants**. Type A 的 inhabitant 就是它的证明 $\blacksquare$，没有证明就是 $\varnothing$. 或者推广到：命题 A 的真值 = A 的 type 作为 集合 的 **cardinality**; The truth valuation of $A = |A|$. 这样看，$A \Rightarrow B$ 的真值 就是 $|B^A|$, 亦即是从 $\{\blacksquare\}$ 或 $\varnothing$ 到 $\{\blacksquare\}$ 或 $\varnothing$ 的 map 的**个数**, 而这个 map 只有在 $\{\blacksquare\} \mapsto \varnothing$ 的时候是空集（不可能）。

这个观察 可以推广到 fuzzy logic (§7.1) 和 strict implication $A \multimap B$ (§6.5).

但 material implication 导致某种悖论：

$$A \wedge B \quad \vdash \quad A \Rightarrow B \tag{2.7.1}$$

例如

$$\text{看见黑猫} \wedge \text{发生车祸} \quad \vdash \quad \text{看见黑猫} \Rightarrow \text{发生车祸} \tag{2.7.2}$$

即任何两件**同时**发生的事件，会导致类似**因果**的结论，然而这因果关系未必成立。这个谬误出现的原因，似乎是因为混淆了不同时间的 **cases**. 换句话说：我们应该验证了 table (2.6.1) 的所有 cases，然后才下结论说 $A \Rightarrow B$;

[*] I learned this from David Corfield's book [**Corfield**]

然而根据 经典逻辑的 material implication，只需要一个 case 就可以下结论说 $A \Rightarrow B$.

那么，这个谬误 为什么没有在 经典逻辑 AI 系统中被发现?

所谓 **inductive learning of logic rules**，其原理是根据以下的「生成模型」:

$$\text{generators} \xmapsto{\text{generate}} \text{data of the world} \tag{2.7.3}$$

这种 generators 的思想， 和 数学中 generators of ideals, groups, function fields, 等 是一样的。

Machine learning 的目的，是求得一组这样的 generators，而它生成的**机制**，即是逻辑推导 $\vdash$.

但由於在 learning 的过程中，需要验证 **不同时间**的 cases，因此避免了 太轻易接受 $A \Rightarrow B$ 的问题。

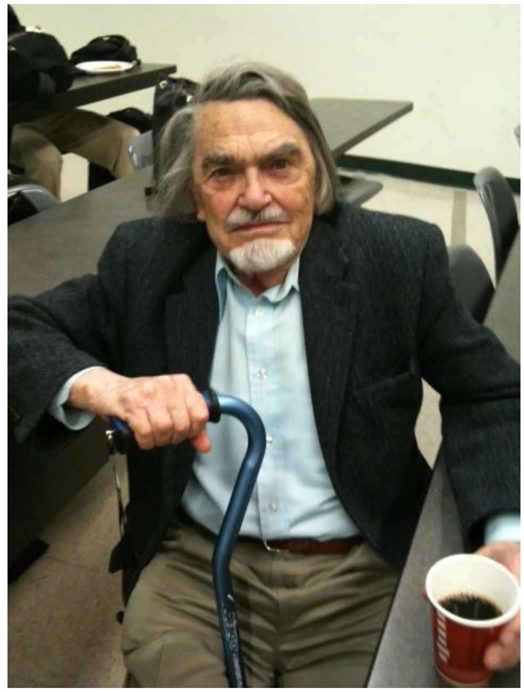
3 Topos theory

3.1 Basics

将 type theory 对应到 category theory，这工作是 Lambek 做的，於是完成了 Curry-Howard-Lambek 的「三位一体」:

$$\begin{tikzcd}[column sep=small] \text{logic} \arrow[rr, dash] \arrow[dr, dash] & & \text{programming} \arrow[dl, dash] \\ & \text{category theory} & \end{tikzcd} \tag{3.1.1}$$

Joachim Lambek (1922-2014)



Topos 的重要意义在於 它是一个可以用来 **进行逻辑运算** 的范畴，关键在於它可以表达 **子集** 的概念，或更一般地叫作 **sub-objects**.

一个 topos $\mathcal{C}$ 里面存在 sub-object classifer Ω 使得 $X \to \Omega \cong$ sub-objects of X. 换句话说 X 的 子集 可以用 $X \to \Omega$ 这个映射来 **represent**.

在 **Set** 这个 topos 里面，Ω 是一个有**两个**元素的集合，可以记作 $\{\top, \bot\}$. 那么 $X \to \Omega$ 就是一些 **命题**，例如 X 是人的集合，则 $X \xrightarrow{\text{mathematician}} \Omega$ 定义哪些人是数学家。

Topos theory 里面最重要的 commutative diagram 是这个：

$$\begin{tikzcd} X \arrow[r, "!"] \arrow[d, tail, "m"'] & 1 \arrow[d, "\text{true}"] \\ Y \arrow[r, "\chi_m"'] & \Omega \end{tikzcd} \tag{3.1.2}$$

其中：

- $X \xrightarrow{!} 1$ 是个 unique arrow，它将 集合 X **整个地**映射到 1. 而 1 是 terminal object，它的定义就是说，通向它的箭咀只能有一个。
- $1 \xrightarrow{\text{true}} \Omega$ 在 $\top$ 和 $\bot$ 之间选择 $\top$, 因此叫 “true” arrow.
- $X \overset{m}{\rightarrowtail} Y$ 是 **monic** arrow，特别地，在 **Set** 里面它就是 **inclusion** map，即 $X \overset{m}{\hookrightarrow} Y$. 它表示 X 是 Y 的**子集**，$X \subseteq Y$.

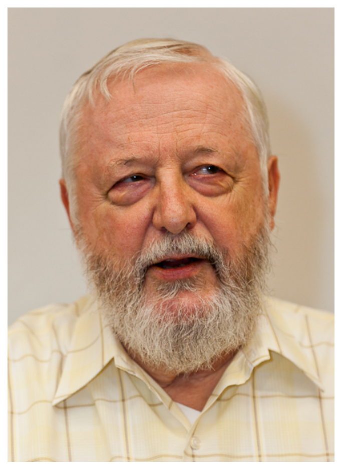
- $Y \xrightarrow{\chi_m} \Omega$ 是集合论中熟悉的 **characteristic function**, 当元素 $e \in X \subseteq Y$ 时，$\chi(e)$ 取值 1，否则为 0. 正是 χ_m 的存在令这幅图 commute. χ_m 也记作「m」.

不熟悉基本范畴论的读者，我非常推荐看一看《Conceptual Mathematics》这本书，写得连中学生也可以看懂，而作者之一的 Lawvere 正是 topos 理论的创始人。

从 **Set** 的角度看，这个 diagram 很易理解，但 topos 的好处是它可以将这些逻辑概念 **generalize** 到比 **Set** 更一般的范畴。

Topos 理论的重要性 在於 它用 category 的语言 **重新表述**了 集合论的整个基础。特别地，逻辑学中的符号，例如 $P(x), \forall x, \exists x$, 表面上看似无法用范畴论表示，这正是 Lawvere 惊人的成就。

William Lawvere (1937-)



再看一次 图 (3.1.2):

$$\begin{tikzcd} X \arrow[r, "!"] \arrow[d, rightarrowtail, "m"'] \arrow[dr, phantom, "\lrcorner", very near start] & 1 \arrow[d, "\text{true}"] \\ Y \arrow[r, "\chi_m"'] & \Omega \end{tikzcd} \tag{3.1.3}$$

右边的 $\begin{array}{c}1\\ \downarrow\\ \Omega\end{array}$ 称为 **generic subobject**. 它的 **pull back** square 用 $\lrcorner$ 记号表示。而左边的 $\begin{array}{c}X\\ \downarrow\\ Y\end{array}$ 则是一般的 sub-object. We say that the **property** of being a sub-object is **stable under pullbacks**.

3.2 Interpreting logic in a topos

Key idea is that propositions are interpreted as "maps towards Ω":

$$p : \Gamma \to \Omega \tag{3.2.1}$$

This is very natural, because propositions are functions that return truth values.

Γ is a **variable context** specifying the arguments (and their types) that the predicate p depends on. For example, if p is of the form $p(x_1, x_2, ...)$, then

$$\Gamma = x_1{:}A_1, x_2{:}A_2, ... = A_1 \times A_2 \times ... \tag{3.2.2}$$

According to topos theory, (3.2.1) is also a **subobject** of Γ. For example, the predicate "Male" is a subobject of the domain of "Humans".

Interpreting $\Rightarrow$

In proof theory, the arrow $\Rightarrow$ arises from the ($\Rightarrow$I) rule:

$$\frac{\Gamma|\Phi, \phi \vdash \psi}{\Gamma|\Phi \vdash \phi \Rightarrow \psi} \qquad (\Rightarrow \mathrm{I}) \tag{3.2.3}$$

where $\Gamma|\Phi$ denotes the **variable context** Γ concatenated with the **propositional context** or **context of assumptions** Φ.

The entailment relation $\vdash$ is interpreted also as subobject or **inclusion**:

$$\Gamma|\ \phi_1, \phi_2, ... \vdash \psi \qquad \text{means} \qquad [\![\phi_1]\!] \wedge [\![\phi_2]\!] \wedge ... \leq [\![\psi]\!] \tag{3.2.4}$$

Recall that, under the Curry-Howard interpretation for propositional logic, $\Rightarrow$ corresponds to λ-terms and to morphisms in a CCC (Cartesian-closed Category). Now every topos is automatically a CCC. In a topos, $\Rightarrow$ is interpreted as sub-objects, but nevertheless they are still morphisms.

3.3 ∀ and ∃ as adjunctions

Let Forms($\bar{X}$) denote the set of formulas with only the variables $\bar{X}$ free. ($\bar{X}$ may contain multiple variables.)

Then one can always trivially add an additional **dummy** variable Y:

$$\delta : \mathrm{Forms}(\bar{X}) \to \mathrm{Forms}(\bar{X}, Y) \tag{3.3.1}$$

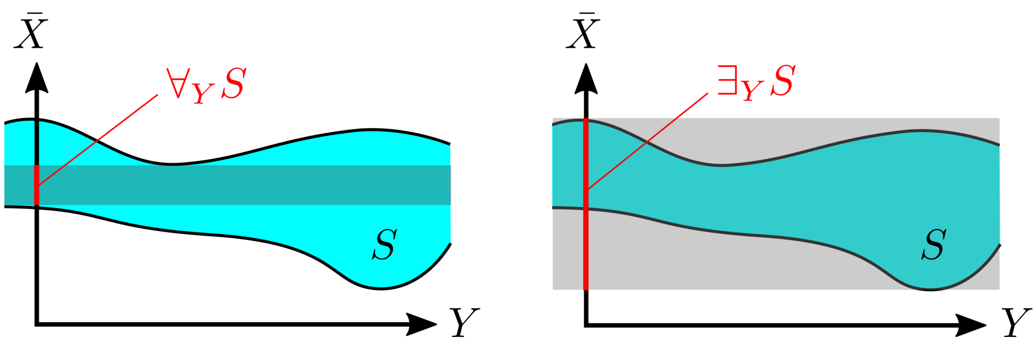
taking each formula $\Phi(\bar{X})$ to itself.

It turns out that $\exists$ and $\forall$ are **adjoints** to the map δ:

$$\begin{tikzcd} \mathrm{Forms}(\bar{X}) \arrow[r, "\delta"] & \mathrm{Forms}(\bar{X}, Y) \arrow[l, shift right=4, "\exists_Y"'] \arrow[l, shift left=4, "\forall_Y"] \end{tikzcd} \tag{3.3.2}$$

or simply denoted as $\exists \dashv \delta \dashv \forall$. 而这是十分 makes sense 的，因为 一个 $\Phi(\bar{X}, Y)$ 的式子，经过 $\forall Y.\, \Phi(\bar{X}, Y)$ 之后，就变成一个和 Y **无关**的式子。

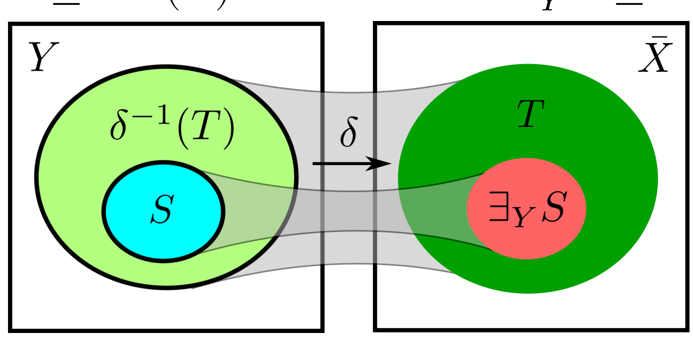
In **cylindric algebra**, the quantifiers $\forall_Y$ and $\exists_Y$ can be interpreted as **projections** where Y is the component that is "killed" by the projections:



(3.3.3)

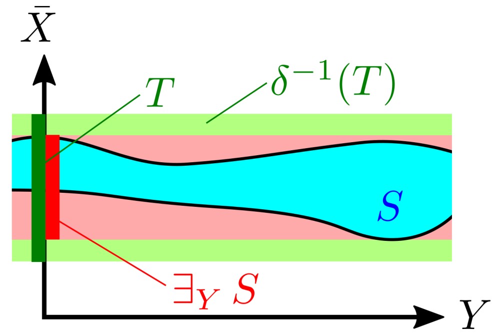
另一个方法是 借助 T 定义 * （注意 在下图中 $\bar{X}$ 和 Y 变成平面）:

$$\forall T \subseteq \bar{X} : \qquad S \subseteq \delta^{-1}(T) \quad \Longleftrightarrow \quad \exists_Y S \subseteq T \tag{3.3.4}$$



* This formula is from [**Abramsky2011**]

熟悉 Galois theory 的读者 可以理解 adjunction 是 **Galois connection** 的推广形式。用日常语言讲：我们知道 S，想定义 $\exists S$；S 存在於论域 Y，$\exists S$ 存在於 论域 $\bar{X}$；借助「影子」T 在 论域 $\bar{X}$ 夹著 $\exists S$，则影子的「原象」在论域 Y 夹著 S. 下图是 δ 是 projection 的特例，δ 是「杀掉」Y 的 projection：

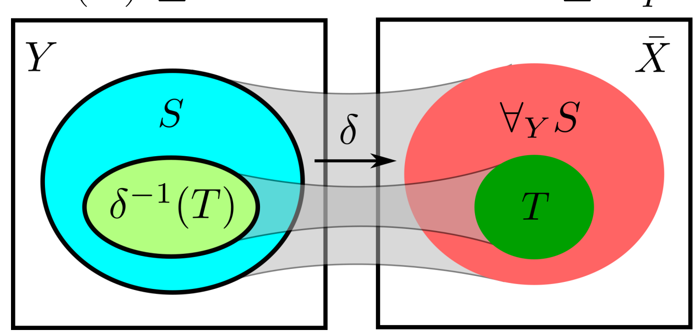


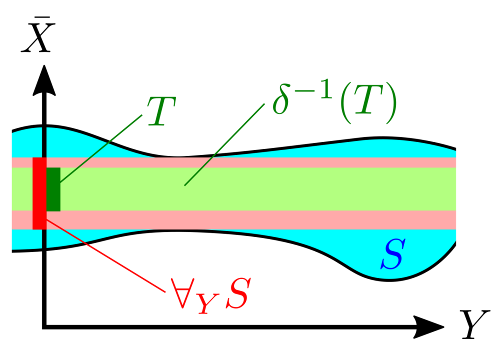
(3.3.5)

注意：(3.3.5) 的 δ 是 $\bar{X} \times Y \to X$ 的 projection，但 (3.3.4) 的 δ 可以是**任何** $Y \to X$ 的映射，这似乎是 $\forall$ 和 $\exists$ 的最一般的定义。范畴论 定义 的好处是方便推广到其他逻辑「模型」，例如 过渡到 Banach space.

类似地有 $\forall$ 的定义：

$$\forall T \subseteq \bar{X}: \qquad \delta^{-1}(T) \subseteq S \quad \Longleftrightarrow \quad T \subseteq \forall_Y S \tag{3.3.6}$$





(3.3.7)

3.4 $\wedge$ and $\Rightarrow$ as product-hom adjunction

一个只有原子命题的逻辑系统是「静止」的，不能推出新的结论：

$$\begin{array}{ccc} A & & A \\ B & \vdash & B \\ C & & C \end{array} \tag{3.4.1}$$

但如果左边加多一个式子 $A \wedge B \wedge C \Rightarrow D$，则可以推出新的结论 D：

$$\begin{array}{rcc} A & & A \\ B & \vdash & B \\ C & & C \\ \color{red}{A \wedge B \wedge C \Rightarrow D} & & D \end{array} \tag{3.4.2}$$

换句话说，$\Rightarrow$ 算符 是逻辑引擎的「燃料」，没有它不能推动逻辑 inference 的过程。

$\vdash$ 的功能和 $\Rightarrow$ 类似，但 $\vdash$ 是 **元逻辑** (meta-logic) 的符号，而 $\Rightarrow$ 是逻辑**之内**的算符。

在 (3.4.2) 中，$\color{red}{A \wedge B \wedge C \Rightarrow D}$ 导致了 $A \wedge B \wedge C \vdash D$ 的出现。

一般来说，$\Delta \Rightarrow \Gamma$ 就是 $\vdash$ 这个映射 对於 Δ 的一个 **截面** (a restriction of the $\vdash$ map to the domain Δ). 这一点很重要：一个 map 作用在某些元素上，但这些元素 和那个 map 是「同类」的。这其实是逻辑结构的一个 defining characteristic.

在 topos 里有一个很重要的 **product-hom adjunction**, 它说的是 $A \wedge B$ 和 $A \Rightarrow B$ 之间的邻接：

$$(A \times B) \to C \quad \simeq \quad A \to (B \to C) \tag{3.4.3}$$

这在逻辑上 是见惯的，没有什么稀奇，它推论 $A \Rightarrow B$ 可以替代 $A \vdash B$. 由於 $\vdash$ 在我们的 AI 系统中是 **神经网络**，这表示 神经网络中的「黑箱」知识可以「外在化」(externalize) 成 **逻辑命题**，这一点 在智能系统中 是有关键的重要性，因为它表示 知识可以透过语言学习得到（虽然这也不是必需 范畴论才可以看得出来。）

3.5 Classifying topos $\leftrightarrows$ internal language

想认识一个**范畴**，最重要的是问：它的 objects 是啥？它的 morphisms 是啥？

Lambek 给出的对应是：

- types $\leftrightsquigarrow$ objects
- terms $\leftrightsquigarrow$ morphisms

We have the following transformations between two formalisms:

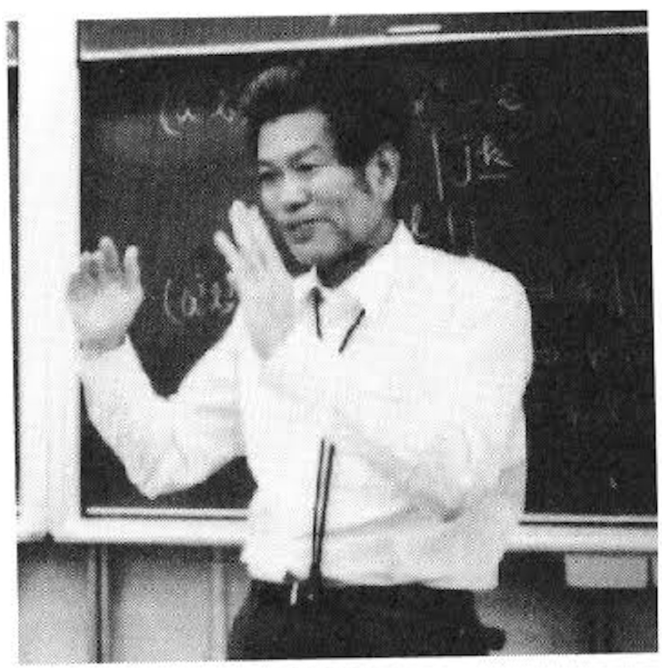
$$\boxed{\text{topos}}\ \mathcal{C} \underset{\text{classifying topos}}{\overset{\text{internal language}}{\rightleftarrows}} T\ \boxed{\text{type theory}}\ . \tag{3.5.1}$$

In other words,

$$\mathcal{C} = \mathcal{C}\ell(T), \quad T = \mathrm{Th}(\mathcal{C}). \tag{3.5.2}$$

3.6 Yoneda lemma

米田 信夫 (1930-1996)



在一个范畴 $\mathcal{C}$ 里面，考虑 其中一个物体 A 到其他物体的 morphism, $A \to \bullet$. 这可以说是，透过 A「看」其他物体的方法。

例： 在 **Set** 里面，$1 \to X$ 是用 终点物体 1「看」其他物体，看到的是集合的**元素**。

例： 映射 $\mathbb{R} \to X$ 是 空间 X 中的 **曲线**，可以说 $\mathbb{R}$「看到」曲线。

例： 在 ordered set $(\mathbb{R}, \leq)$ 里面 物体 $0 \to x$ 可以「看到」x 是不是 **positive**.

类似地，可以考虑 对偶 的情况，$\bullet \to A$ 是其他物体怎样「看」A 的方法。

例： 在 **Set** 里面，$X \to 2$ 是其他物体「看」2 的方式，得到的是 X 的**子集**, $\wp(X)$.

例： 在 **Top** 里面，2 包含一个 open set 和一个 closed set，$X \to 2$ 得出的是 X 的**开子集**，$\mathrm{Opens}(X)$.

A functor $X : \mathcal{A} \to \mathbf{Set}$ is **representable** if $X \cong H^A$ for some $A \in \mathcal{A}$.

H^A 的意思是 $\mathcal{A}(A, -)$, 这是一个集合。

A **representation** of X is a choice of an object $A \in \mathcal{A}$ and an isomorphism between H^A and X.

Yoneda embedding of $\mathcal{A}$:

$$H_\bullet : \mathcal{A} \to [\mathcal{A}^{\mathrm{op}}, \mathbf{Set}] \tag{3.6.1}$$

$$\begin{aligned}
\text{For each } A \in \mathcal{A}, \text{ we have a functor} \quad & \mathcal{A} \xrightarrow{H^A} \mathbf{Set} \\
\text{Putting them all together gives a functor} \quad & \mathcal{A}^{\mathrm{op}} \xrightarrow{H^\bullet} [\mathcal{A}, \mathbf{Set}] \\
\text{For each } A \in \mathcal{A}, \text{ we have a functor} \quad & \mathcal{A}^{\mathrm{op}} \xrightarrow{H_A} \mathbf{Set} \\
\text{Putting them all together gives a functor} \quad & \mathcal{A} \xrightarrow{H_\bullet} [\mathcal{A}^{\mathrm{op}}, \mathbf{Set}]
\end{aligned} \tag{3.6.2}$$

$$\begin{tikzcd} \mathcal{A}^{\mathrm{op}} \arrow[r, bend left, "H_A"] \arrow[r, bend right, "X"'] \arrow[r, phantom, "\Downarrow"] & \mathbf{Set} \end{tikzcd} \tag{3.6.3}$$

Yoneda lemma:

$$[\mathcal{A}^{\mathrm{op}}, \mathbf{Set}](H_A, X) \cong X(A) \tag{3.6.4}$$

{ How sheaves give rise to representables.... }

Application: continuation-passing style

There is an isomorphism between the types a and $\forall r.(a \to r) \to r$. It follows from the Yoneda lamma.

In the theory of functional programming, the type $\forall r.(a \to r) \to r$ is also known as the **continuation-passing style** (CPS) of the type a.

Intuitively, CPS can be understood as saying that having a value is just as good as having a function that will give that value to a callback.

CPS is of special interest to logic, because it allows to extend the Curry-Howard correspondence from intuitionistic logic to **classical** logic.

{ Explain CPS in functional programming... }

There exists a **Kolmogorov translation** $k(\cdot)$ that converts a proposition such that (basically) each atom is preceded by a double negation ($\neg\neg$). This translation has the property that

$$\phi \text{ provable in classical logic} \Leftrightarrow k(\phi) \text{ provable in intuitionistic logic.} \tag{3.6.5}$$

The Kolmogorov translation is a special case of CPS.

(Application?) symmetric neural networks

(我觉得这个可能是 Yoneda lemma 的一个应用，但根据 MathOverflow 上一些专家的看法，只是表面上有点相似。

The **Kolmogorov-Arnold representation theorem** states that every multivariate continuous function can be represented as a sum of continuous functions of one variable:

$$f(x_1, ..., x_n) = \sum_{q=0}^{2n} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right) \tag{3.6.6}$$

It can be specialized to such that every symmetric multivariate function can be represented as a sum of (the same) functions of one variable:

$$f(x_1, ..., x_n) = g(h(x_1) + ... + h(x_n)) \tag{3.6.7}$$

Cayley's theorem: Any group can be represented as a sub-group of a special group, namely the permutation group.

Any symmetric function can be represented as a sub-function of a special symmetric function, namely the sum.

3.7 Model theory, functorial semantics

Model theory is basically a **functorial** map from logic formulas to algebraic objects:

$$a \cdot b \longmapsto [\![a]\!] \cdot [\![b]\!] \tag{3.7.1}$$

On the left side is a syntactic formula (eg. logic formula), and on the right side is an object composed of elements in an algebraic **language** or **structure**.

For example, if we want to define an **Abelian group**, we would have a (syntactic) axiom that says:

$$a \cdot b = b \cdot a \tag{3.7.2}$$

which expresses a formal (syntactic) rule concerning its symbols. Using model theory, the syntactic object such as $a \cdot b$ is mapped to the actual algebraic objects, ie, the elements of an Abelian group. Crucially, the multiplication symbol $\cdot$ is also mapped to the group multiplication operation. We say that such a mapping is **functorial** — it maps elements to elements and operations to operations.

This is the main idea behind functorial semantics.

We interpret formulas in a topos $\mathcal{E}$ by assigning each an **extension**. In topos theory this is called **internal** semantics. Note the terminology is a bit confusing.

3.8 Generalized elements

一个 逻辑命题 ϕ 可以看成是由 某论域 $A \xrightarrow{\phi} \Omega$ 的函数，其中 $\Omega = \{\top, \bot\}$.

也可以说：命题 $\phi(x)$ 是真的，其中 x 是 A 的**元素**。In category theory, we use the terminal object $\mathbf{1}$ to "pick out" elements of A, as follows:

$$1 \xrightarrow{x} A \xrightarrow{\phi} \Omega. \tag{3.8.1}$$

In **Set**, 任意一个由 1 出发的函数 $x : 1 \to A$ 可以直接看成是 A 的「元素」。

但如果我们用另一个 论域 C 取代 1，换句话说：

$$C \xrightarrow{x} A \xrightarrow{\phi} \Omega. \tag{3.8.2}$$

这样的 $x : C \to A$ 叫作 A 的 **generalized element**.

另一个术语是：C **forces** $\phi(x)$, notation: $C \Vdash \phi(x)$. 这术语来自 Paul Cohen 为了解决 Continuum Hypothesis 而提出的 forcing 技巧，见 §3.11.

另一个说法是：$\phi(x)$ is true **at stage** C.（这术语来自 possible-world semantics）

3.9 Internal vs external semantics

假设 $\phi(\bullet)$ 是一个谓词，ϕ 的论域 (domain) 是 A. 例如 $\phi(x)$ 表示 x 是男性，$x \in$ 人。则 $\phi(x)$ 这个命题的 **extension**（外延）就是在「人」的集合中属於「男性」的元素。

知道了 ϕ 的 extension 就可以判断，对每一个 $x \in A, \phi(x)$ 的真假。换句话说，提供了一种 **interpretation** 的方法。

The "internal" way to interpret type theory in a topos is where a formula ϕ in context $x_1 : A_1, ..., x_n : A_n$ is interpreted as a **subobject** of $A_1 \times ... \times A_n$.

这个方法叫 **internal semantics**.（注意 extension 的概念是 internal 的，有点混淆）

另一种方法是：指定 哪些 generalized elements 符合某个 谓词。后者叫 **external semantics**：

3.10 Kripke-Joyal / external semantics

External semantics describe which generalized elements satisfy each formula.

Generalized element 的意思是 $I \xrightarrow{a} A \xrightarrow{\phi} \Omega$, 记作 $a \Vdash \phi$.

A generalized element satisfies a formula iff it is a member of the formula's **extension**.

3.11 Cohen's method of forcing

在 数理逻辑 / 集合论 里，forcing 指的是一些 加在某 集合 G 上的 条件，说某些 元素 属於 或不属於 G. 这些 条件 用逻辑表达，例如：

$$c = \{3 \in G, 57 \notin G, 873 \notin G\} \tag{3.11.1}$$

is a condition.

例如，条件 c 可以「强迫」集合 G 里面至少有 100 个质数、或至少有 1 万个质数。记作 $c \Vdash P$, P 是某个逻辑命题。

以下内容和 AGI 无关，但因为数学上有趣所以写一下。

Continuum hypothesis (CH):

$$2^{\aleph_0} = \aleph_1 \tag{3.11.2}$$

这是说：连续统 $[0,1]$ 的基数 $2^{\aleph_0}$ 紧接在 可数集合的基数 之后。

1878 年，Cantor 提出 CH
1900 年，Hilbert 列出 连续统假设 为「23 问题」的第一个
Hilbert 给出了一个证明，但里面有 bug
1938 年，Gödel 证明 ZF + CH is consistent，换句话说：ZF cannot disprove CH
1963 年，Paul Cohen 证明 ZF cannot prove CH
他用的方法叫 “forcing”

Paul Cohen (1934-2007)

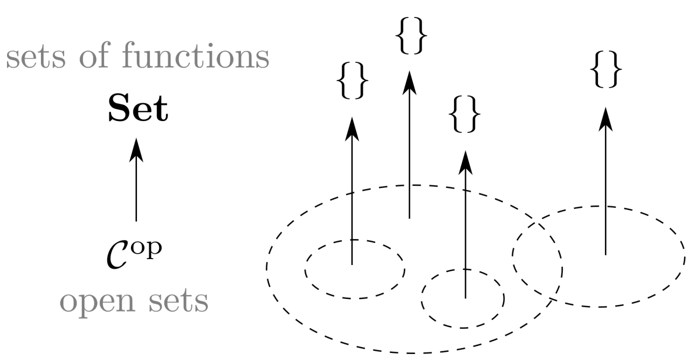
3.12 Sheaves

Sheaves are automatically toposes, and like the topos they capture the idea of "indexing".

Functors $\mathcal{C}^{\text{op}} \to \mathbf{Set}$ are called **pre-sheaves** on $\mathcal{C}$.

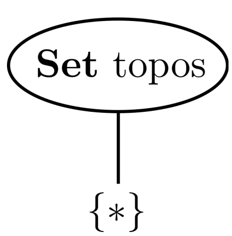
Pre-sheaf 要变成 sheaf 还需符合 sheaf condition，这是一种典型的 gluing condition, 意思是在 $U_i \cap U_j$ 这些重合的邻域上，presheaf 的 sections 是**一致**的。

当 $\mathcal{C}^{\text{op}}$ = topology of open sets 时，我觉得下图特别容易理解：



(3.12.1)

As a special case, the sheaf over the singleton $\{*\}$ is the **Set** topos:



(3.12.2)

In general, a sheaf is built upon a **site** which is a topological space with underlying set X and its set of opens $\mathcal{O}(X)$. The subobject classifier Ω depends on an open set U chosen from $\mathcal{O}(X)$. In general, Ω is the "opens of U". For example, in the special case of the **Set** topos above, Ω is the opens of $\{*\}$ which is either $\varnothing$ or $\{*\}$. Therefore the Ω in **Set** is $\{0, 1\}$.

The sheaves defined in this way are automatically toposes and they are known as **Grothendieck toposes**.

Some **Set**-valued functors are **representable**, ie, isomorphic to a hom-functor.

For an object S of a category $\mathcal{C}$, the functor

$$H^S : \mathcal{C} \to \mathbf{Set} \tag{3.12.3}$$

sends an object to its set of generalized elements of shape S. The functoriality tells us that any map $A \to B$ in $\mathcal{C}$ transforms S-elements of A into S-elements of B. For example, taking $\mathcal{C} = \mathbf{Top}$ and $S = S^1$, any continuous map $A \to B$ transforms loops in A into loops in B.

In logic, the category of predicates can be regarded as a sheaf over its domain $\begin{array}{c}\mathbf{Pred}\\ \downarrow\\ \mathbf{Set}\end{array}$.

我暂时还不太清楚 sheaf 理论对 AI 来说有什么重大意义。

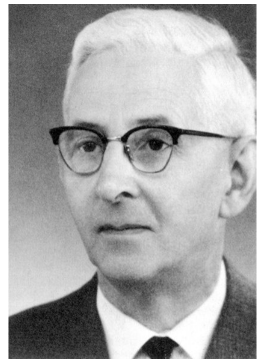
3.13 Kleene realizability

4 Intuitionistic logic

In 1933, Gödel proposed an interpretation of intuitionistic logic using possible-world semantics.

In topos theory $A \Rightarrow B$ is adjoint (via the hom-product adjunction) to $A \vdash B$, which is "okay" because it is independent of which implication (material or strict) we are using.

4.1 Heyting algebra

Arend Heyting (1898-1980)  (4.1.1)

1930 年，Heyting 给出了 constructive mathematics 的一种 axiomatization, called **intuitionistic logic** (IL). Heyting algebra 是一种 IL 的 **代数模型**，正如 Boolean algebra 是 经典逻辑的 代数模型。(Heyting algebra is to intuitionistic logic what Boolean algebra is to classical logic.)

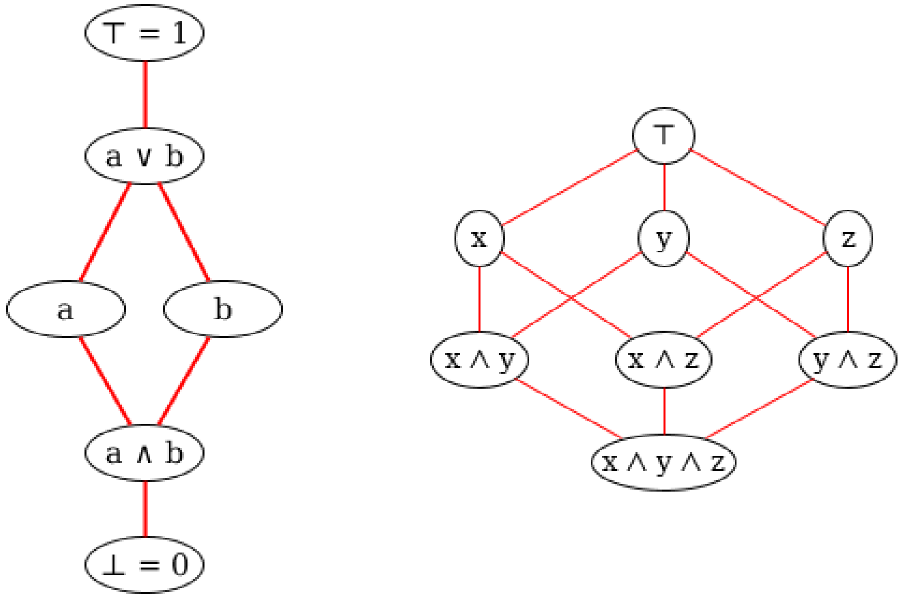
Reference: [**Dunn2001**]

A lattice is an **order** structure. There are many variations of order structures, satisfying various conditions such as distributivity, modularity, etc, which we won't go into details. Among lattice structures, the examples important to us are: **Boolean lattices** (which model propositions of classical logic) and **Heyting lattices** (which model propositions of intuitionistic logic).

The important thing to remember is: in lattice theory, the order $a \leq b$ usually corresponds to $a \Rightarrow b$ in logic. Notice the arrow direction is reversed [*]. In the lattice, we can form new elements such as $a \wedge b$ and $a \vee b$, these of course correspond to compound logic propositions formed via logic connectives. The lattice shows us the **implication order** ($\Rightarrow$) among such propositions. For example, the following

[*] I don't know the precise reason for this, perhaps this is just a matter of convention.

are Boolean lattices of 2 and 3 elements [*] :

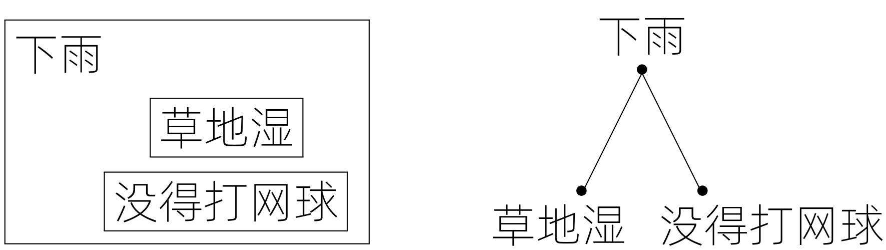


(4.1.2)

The above are examples of **Hasse diagrams**, with edges showing the $\leq$ order. Notice that not every $\leq$ relation is shown; the Hasse diagram only shows those edges where $a \leq b$ "immediately", ie. the "cover" relation:

$$a \text{ covers } b \quad \text{means} \quad a \leq b \text{ "immediately"} \tag{4.1.3}$$

以下的 **拓樸**模型 和 **格** (lattice) 模型，都是 Heyting algebra 的模型：



(4.1.4)

两者之间是等价的，起源於 **Stone duality** (every Boolean algebra is isomorphic to a topology of open sets)，其后再被推广到 **Priestley** 拓扑对偶 等，都是大同小异的。

A lattice can be equivalently turned into an algebra via:

$$a \leq b \quad \Leftrightarrow \quad a * b = a. \tag{4.1.5}$$

[*] I have not carefully examined why some elements are included or not.

This yields a semi-lattice. A lattice has 2 operations, where $*$ can be either meet $\wedge$ or join $\vee$.

注意：下雨 $\rightarrowtriangle$ 草地湿 是 Heyting implication，这个 $\rightarrowtriangle$ 的存在 并不是因为「下雨」的**真值** 比「草地湿」的**真值 小**。

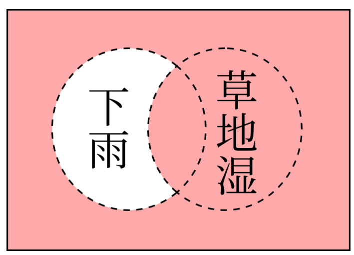
The Heyting implication $a \rightarrowtriangle b$ exists for all elements a, b, x such that:

$$x \leq (a \rightarrowtriangle b) \quad \text{iff} \quad (x \wedge a) \leq b. \tag{4.1.6}$$

Every Boolean algebra can be a Heyting algebra with the material implication defined as usual: $a \Rightarrow b \equiv \neg a \vee b$.

可以用以下例子说明 Heyting implication $\rightarrowtriangle$ 的定义：

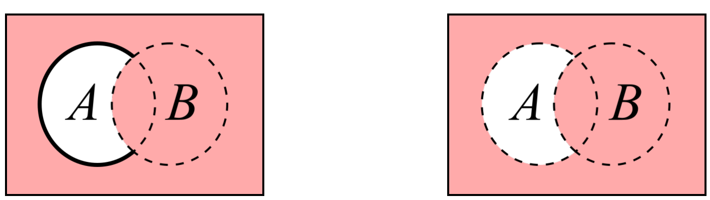
$$\forall X. \quad X \subseteq (\text{下雨} \rightarrowtriangle \text{草地湿}) \quad \text{iff} \quad (X \wedge \text{下雨}) \subseteq \text{草地湿} \tag{4.1.7}$$



(4.1.8)

经典逻辑的 $\Rightarrow$ 和 直觉主义逻辑 $\rightarrowtriangle$ 只有 在**边界**上的 微小差异：

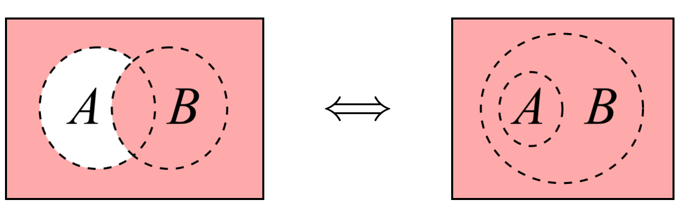
classical $A \Rightarrow B$ Heyting $A \rightarrowtriangle B$



(4.1.9)

无论是何种逻辑，以下的两边是 **等价**的：

$$A \Rightarrow B \quad \Longleftrightarrow \quad A \vdash B \text{ or } A \subseteq B \tag{4.1.10}$$



(4.1.11)

这可以看成是 将白色部分不断 缩小并「消灭掉」的结果。

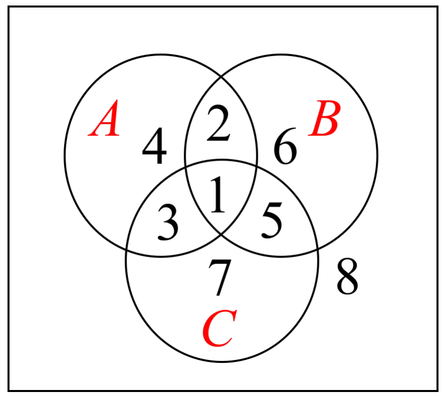
根据 Stone duality, 既然可以用 Boolean 或 Heyting algebra 做逻辑推导，因此也可以用 toplogy of (open) sets 做逻辑推导。其根据就是 以下的等价：

$$A \vdash B \quad \Leftrightarrow \quad A \subseteq B \tag{4.1.12}$$

这种 topology 可以视为 **truth table** 的一种形式，如下图所示，每个**区域**代表 真值表的**一行**：

A	B	C	#
$\top$	$\top$	$\top$	1
$\top$	$\top$	$\bot$	2
$\top$	$\bot$	$\top$	3
$\top$	$\bot$	$\bot$	4
$\bot$	$\top$	$\top$	5
$\bot$	$\top$	$\bot$	6
$\bot$	$\bot$	$\top$	7
$\bot$	$\bot$	$\bot$	8

$$\Longleftrightarrow \tag{4.1.13}$$



之所以讲这么多，目的是想说明，Heyting algebra 和我们熟悉的 Boolean algebra 其实有很多共通点。

另一点发现是，在 §2.7 所说的 material implication 问题，在 intuitionistic logic 里仍然存在。

In a topos $\mathbb{E}$, the subobject $\mathrm{Sub}_{\mathbb{E}}(A)$ is a **poset** that admits **Heyting implication**.

Using Kripke semantics, the Heyting arrow $\rightarrow$ can be defined by:

$$k \Vdash A \rightarrow B \quad \Leftrightarrow \quad \forall \ell \geq k \; (\ell \Vdash A \Rightarrow \ell \Vdash B) \tag{4.1.14}$$

Whereas the "fish-hook" **strict implication** can be defined as "A implies B necessarily":

$$A \multimap B \quad \equiv \quad \Box(A \Rightarrow B) \tag{4.1.15}$$

The two can be regarded as equivalent via:

$$\begin{aligned} k \Vdash \Box(A \Rightarrow B) \quad &\Leftrightarrow \quad \forall \ell \geq k \; (\ell \Vdash (A \Rightarrow B)) \\ &\Leftrightarrow \quad \forall \ell \geq k \; (\ell \Vdash A \Rightarrow \ell \Vdash B) \end{aligned} \tag{4.1.16}$$

问题是将 Heyting implication 定义 by possible worlds semantics 有什么好处? 从 machine learning 角度可能更合理。但其实也好像包含 classical implication 所以是循环的?

5 Logic's connections with algebras

[This section will be expanded later...]

In the book *"Term Rewriting and All That"* [1998, Ch. 8], it is mentioned that Huet's **completion procedure** for rewriting logic is roughly the same as **Buchberger's algorithm** for finding Gröbner bases for polynomial rings. Huet's completion procedure takes as input a set of rewriting rules and tries to find an equivalent rewriting system that is **convergent** (ie. guaranteed to terminate), or return failure. In computational algebra, Gröbner bases are used to decide the **ideal congruence** and **ideal membership** problems in polynomial rings. The ideal congruence problem can be seen as a **word problem**, and Gröbner bases define convergent reduction relations on polynomials.

There is another, perhaps related, and perhaps even more important connection: If we apply the Curry-Howard correspondence to "Hilbert-style" proofs, we find that proof elements behave as **combinators**, as in combinatory logic. This gives rise to **BCI algebra**, a kind of non-associative algebra capturing the syntax of $\rightarrow$. The name BCI comes from the combinators B, C, I.

Using techniques from **representation theory**, one starts with a **quiver** Q, which is a kind of graph, and gets a **path algebra** kQ and its vector-space **representation**. The path algebra is a vector space with all possible paths (of lengths ≥ 0) forming its basis, with multiplication given by path concatenation. For its representation, each vertex is seen as a vector space and each path is a linear transformation between the spaces.

With matrix representations of logic inference, maybe we can perform (syntactic) inference with matrices? Inference would be performed as multiplication of matrices. This is very different from the "mainstream" approach, where syntactic inference is implemented as a complicated mapping by a neural network.

6 Modal logic

Modalities are often conceived in terms of variation over some collection or **possible worlds**.

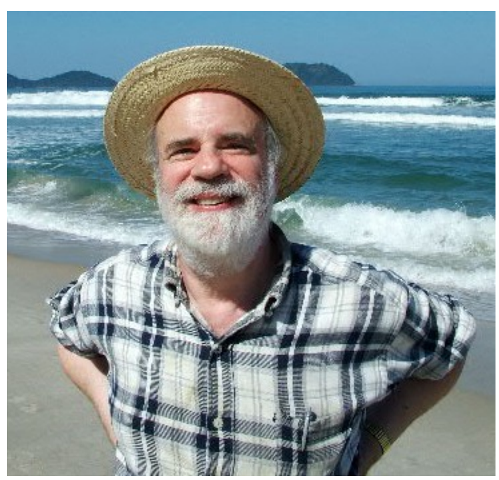
A modal operator (such as $\Box$) in the category $\mathbf{Sheaf}(X)$ is a **sheaf morphism** $\Box : \Omega \to \Omega$ satisfying 3 conditions, $\forall U \subseteq X$ and $p, q \in \Omega(U)$:

$$\begin{aligned} &\text{a)} \quad p \leq \Box(p) \\ &\text{b)} \quad (\Box;\Box)(p) \leq \Box(p) \\ &\text{c)} \quad \Box(p \wedge q) = \Box(p) \wedge \Box(q) \end{aligned} \tag{6.0.1}$$

6.1 Possible-world semantics

Possible-world semantics is also called **intensional semantics**, as opposed to **extensional semantics** where truth values are directly assigned to propositions. Does this idea jibe with the other definition of "intension", ie, as opposed to Leibniz extensionality and also related to intensional logic?

Saul Kripke (1940-)



6.2 Computer implementation of possible worlds

要詮釋 modal logic，需要引入 frame $F = \langle W, R, D, H \rangle$, 其中：

- W = set of possible worlds = $\{w_1, w_2, ...\}$
- R = a relation between worlds, $w_i R w_j$
- D = domain of first-order objects
- $H : W \to \wp(D)$, for each world specify a subset of objects

To interpret formulas with $\Box$:

$$M \Vdash \Box A\ [w] \quad \Leftrightarrow \quad \forall w' \succeq w.\ M \Vdash A\ [w'] \tag{6.2.1}$$

這牽涉到要 quantify over all w's.

重點是 inference 需要什麼 data?

顯然需要決定在某 w 中 p 是否成立，甚至需要判斷 $\forall w.p[w]$.

後者似乎需要將所有 有關的可能世界（或至少是其 summary）放到 working memory 再 quantify.

6.3 Intensional vs extensional

"Beethoven's 9th symphony" and "Beethoven's choral symphony" have the same **extension** but different **intensions**.

6.4 Intensional logic

Possible-world semantics is also called **intensional semantics**, as opposed to **extensional semantics** where truth values are directly assigned to propositions.

Logic terms differ in intension if and only if it is **possible** for them to differ in extension. Thus, **intensional logic** interpret its terms using possible-world semantics.

6.5 Strict implication

The problem of "material implication"

Material implication 的意思是「实质蕴涵」，亦即是说 $A \Rightarrow B$ 等价於 $\neg A \vee B$，其真值表如下：

A	B	$A \Rightarrow B$
0	0	1
0	1	1
1	0	0
1	1	1

(6.5.1)

Material implication 的概念向来很有争议，例如，当前提是错误时，它永远是真的：

$$\text{瑞士在非洲} \Rightarrow \text{猪会飞} \tag{6.5.2}$$

透过观察 truth table 可以发现，它的每一列 其实代表一个 **可能世界**，这些 可能世界 是不会 同时发生的。换句话说，material implication 和 strict implication 本来是一样的，只是前者将 可能世界 的语义 隐蔽到「幕后」。

For strict implication to make sense, it is always necessary to invoke possible-world semantics. A strict implication is always **learned** from numerous examples from experience, in accord with the philosophical tradition of "empiricism".

Strict implication is equivalent to material implication over multiple instances. The truth table of material implication agrees with the functional interpretation of implication.

以下部分 包含很多未落实的想法，包括我自己的 "thinking aloud"

English translation proofread up to here.

7 Fuzzy logic

Lotfi Zadeh (1921-2017)
[Iranian-Jewish]

To develop a "fuzzy topos" theory, one requirement is to make the following diagram commute:

$$\begin{tikzcd} X \arrow[r, "!"] \arrow[d, tail, "m"'] & \mathbb{1} \arrow[d, "\zeta"] \\ Y \arrow[r, "\chi_m"'] & \Omega \end{tikzcd} \tag{7.0.1}$$

where the subobject classifier Ω is now a **lattice**. ζ is a fuzzy "distribution". This commutative diagram is similar to the one for **moduli space**, where every family over a base scheme B arises from the **pullback** of an identity map $\overset{\mathcal{C}}{\underset{\mathcal{M}}{\downarrow\mathbb{1}}}$ where $\mathcal{M}$ is the moduli space:

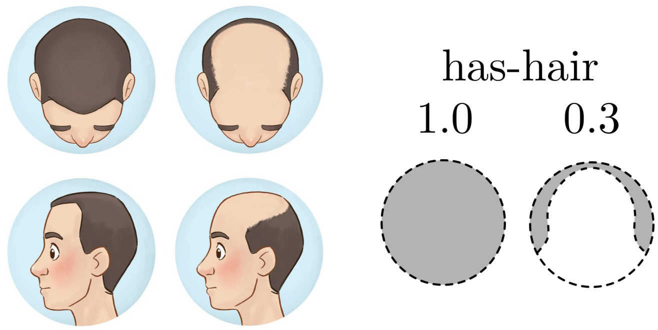
$$\begin{tikzcd} \mathcal{D} \arrow[r] \arrow[d, "\phi"'] & \mathcal{C} \arrow[d, "\mathbb{1}"] \\ B \arrow[r, "\chi"'] & \mathcal{M} \end{tikzcd} \tag{7.0.2}$$

The unit interval $[0, 1]$ as Ω can be understood as a lattice, ie, ordered open sets:

$$\varnothing \subseteq \updownarrow \subseteq \big\updownarrow \subseteq \Big\updownarrow \subseteq \bigg\updownarrow \subseteq \Bigg\updownarrow \subseteq \Bigg\updownarrow \subseteq \Bigg\updownarrow \subseteq \Bigg\updownarrow\,{}^{1}_{0} \tag{7.0.3}$$

This is a category with $\subseteq$ as the $\leq$ morphism.

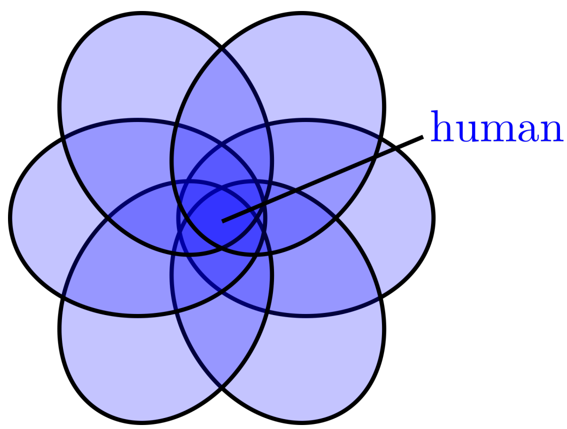
Even more figuratively, we can actually think of the human head with a topology of open sets:



(7.0.4)

This gives a very intuitive interpretation of the fuzzy degree: as the **ratio** of the has-hair area to the full area.

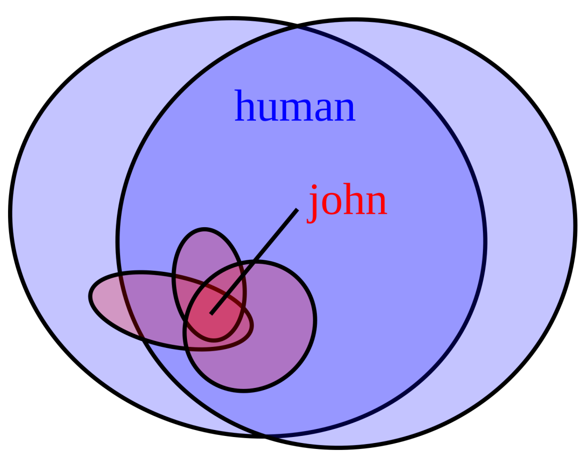
举例来说，「人」的概念由若干个 **属性** (properties) 表示：



(7.0.5)

例如说：人是动物的一种、人有智慧、人有某些感情、人有某些道德观念、等。这些属性是「人」这个概念的 **intension**. 但「人」这个概念的 **extension** 是这个 **集合**的**元素**，例如 John, Mary 等。

而「John」的属性是更为 **特殊** (specific) 的：



(7.0.6)

问题是怎样决定 "John is human" = human(john) 这个命题的 fuzzy truth value.

或者用一个男人更容易明白的例子：当我们说「玛丽莲梦露 很性感」，这个命题的 fuzzy truth value 就是「Marilyn 的属性」和「性感的属性」之间的某种**共同性**的量度。这种 truth value 的定义 是基於 **intension** 而不是 extension.

但 dually，用 **extension** 的角度定义 truth value 也未尝不可：

- 每个 fuzzy proposition 看成是一个 set, 里面有 该命题的「证明」
- 例如「人类」的集合是「有人类」这个命题的证明
- 而，「数学家」作为「人类」的子集，等於命题「所有人都是数学家」的证明

但在某些情况下，"A is B" 但 A 不是**集合**（例如 John 或 Marilyn 不是集合），则 extensional 的诠释 不适用。

无论 intensional 或 extensional, 这种诠释方法 跟 HoTT 理论 (§8) 是融洽的：概念「John」的 **证明** 是 $J_1 \wedge J_2 ... \wedge J_n \Rightarrow$ John, 其中 J_i 是 John 的 **属性**。同理，「handsome」的证明是 $H_1 \wedge H_2 ... \wedge H_n \Rightarrow$ handsome. 因此 "John is handsome" 的 fuzzy truth value 是 $\{J_i\}$ 和 $\{H_i\}$ 的共通性（例如交集或并集）的某种量度。

关於 fuzzy truth value 还有个问题：一个对象的 **属性** 在某程度上可以随意增加或减少，因此纯粹 计算 属性的**数量**，并不能客观地决定 truth value. 似乎每个 属性 应该有某种 **测度** (measure) 或 **权重** (weight), 这样在改变 属性 的描述时，fuzzy truth value 保持**不变**。

问题是 witness 是什么？例如「匙羹在杯内」是因为其他 视觉 propositions 得到的结论。它是 true 这一点是重要的，但它的 "intension" 更重要。或者说 proof 过程中处理的是 proof objects.

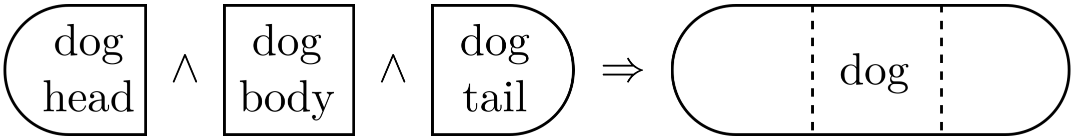
这个 proof object 它除了是一件 syntactic 的东西之外还可以是什么？或者说 map 的 **domain** 是命题，但被 map 映射的是 evidence?

7.1 Fuzzy implication

Consider this (rather simplistic) **binary** logic formula for recognizing a "dog":

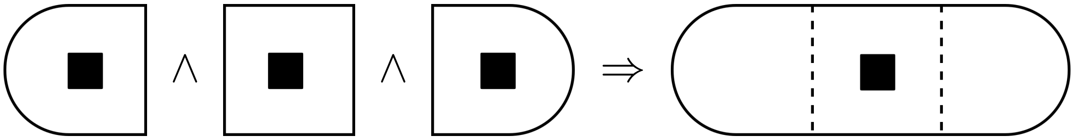
$$\text{dog-head} \wedge \text{dog-body} \wedge \text{dog-tail} \Rightarrow \text{dog} \tag{7.1.1}$$

From the Curry-Howard view, each atomic proposition corresponds to an abstract space representing its type:



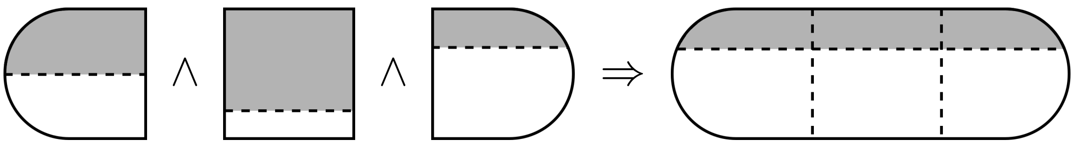
(7.1.2)

In binary logic, each type-space is either inhabited or empty:



(7.1.3)

We want to extend this to the fuzzy case. Each proof will have a **fuzzy degree** which is an element from the $[0, 1]$ lattice, which we represent by sub-spaces of the type-spaces:



(7.1.4)

where we interpret $\wedge$ as min(), so the fuzzy truth value of the conclusion is the minimum among the antecedents. The above picture seems compatible with Voevodsky's HoTT, where fuzzy sets are at level 0, and the identity of elements are expressed by propositions (level -1). Indeed, the fuzzy type-space may have a metric structure allowing to **measure** areas.

本节 考察一下 fuzzy implication $A \tilde{\Rightarrow} B$ 的真值是如何确定的？在 §2.7 已经分析过 经典 material implication 的性质。

From above analysis, material implication seems to be a statistical result. Some cases cause the implication to be true, some cases cause it to be false. Overall, it's truth value would have to be consistent with **all** cases; which (under binary logic) is feasible.

So, is this saying that the TV of (binary) implication is statistically determined? If so, can we just do the same for fuzzy implication? Sure, why not?

But then do we still have a "truth table" of fuzzy implication? What exactly is a "truth table", anyway?

If $A \tilde{\Rightarrow} B$ is a proposition, then what are its **proofs**? This should be interpreted as "the number of maps from A to B". 关键是利用这一点来定义 fuzzy implication 的 truth value.

Fuzzy truth values arise from imperfect proofs. Fuzzy implication arises from maps between imperfect proofs. Why does a map between proofs become less than 1?

首先，number of maps 其实都有问题。根据 fuzzy intension，其实不是 number of proofs 而是 proof 的 imperfect. The measure of whether one proof is a subset of another proof. This forms the basis of a "predication" proposition. But what if the proposition is an implicational one? Then it would be a map from the proof of one proposition to the proof of another.

Rethink the binary case. There exist (or not) proofs of A, B. From which we establish a map from proof of A to proof of B. But when the proof of A is empty then there could be no such map. The truth value of the implication statement must somehow be **consistent** with the **existence** of such maps. So there is a case where such a map is impossible and thus the implication would be false.

如果 A 的 proof fuzzy 到几乎唔存在，（而 B 的 proof is true），咁 $A \Rightarrow B$ 的 fuzzy TV 是不是应该接近 false? Well, there is the possibility of probabilistic. Suppose we assume that it is deterministic but fuzzy. Then the above requirement is clearly correct (if we just think of a simple example). So the fuzzy implication's TV is just a generalization of the binary **truth table**.

But how can we interpret the fuzzy implication's TV as a measure (perhaps the cardinality) of its **map**? According to the binary view, it is a measure of the existence of maps between A and B. In which case, a weak proof of A means that the map from A to B would be weak also.

Looking at the following table from a "conditional" view, it seems reasonable:

A	B	$A \overset{\sim}{\Rightarrow} B$	B^A
weak	weak	strong	$0^0 = 1$
weak	strong	strong	$1^0 = 1$
strong	weak	weak	$0^1 = 0$
strong	strong	strong	$1^1 = 1$

(7.1.5)

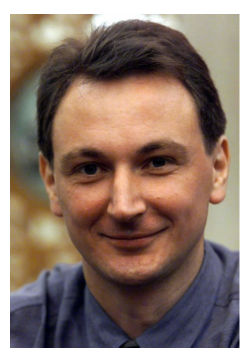
but the problem is how to interpret the truth value of $A \overset{\sim}{\Rightarrow} B$ as **counting** the number of maps?

7.2 Fuzzy functions?

What are fuzzy functions? 

8 Homotopy type theory (HoTT)

Vladimir Voevodsky (1966-2017)



HoTT 的中心思想是将 types 看成是某些 空间 (spaces)，而这些空间可以被赋予 topological 特别是 homotopy 结构。在 homotopy 而言，关键是 将 type A 里面两个元素的 **相等** id_A 看成是 homotopy 的 **path**.

8.1 Why HoTT may be useful

在 topos 上 $A \subseteq B$ 或 $a \in A$ 是一种 subset 关系，或者可以说是 topological 关系。在计算机上，将这个 topology 赋予更多「空间」的特性，例如 vector space, metric space 等，似乎会是很 powerful 的。

但问题是：如果 A 是 拓扑空间中某个（形状已知的）region，而 entity a 的位置也是知道的，则 $a \in A$ 这个命题的真假 也立即可以知道。但这是非常有问题的，例如「$e \in ?$ 无理数」这个命题，必需复杂的证明，而不是瞬间可以判断。又或者「OJ Simpson $\in ?$ 杀人凶手」。

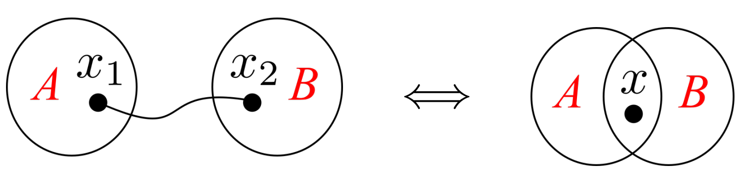
换句话说，似乎不可能将 命题空间 实现成某种有 **具体位置**的空间。因此 $a \in A$ 这种命题，似乎 只能够表述成 ordered pair (a, A)，然后用**代数方法** 模拟 abstract topology. 这 其实就是我在 (9.2.1) 提到的「粗暴」syntactic 方法。

挽救「位置空间」的一个办法是：容许 OJ Simpson $\in$「黑人、男性、演员、球员」等，但保证他不属於其他（未被验证的）集合，后者是**未知**的。但这样导致要 验证 潜在无穷多个集合，并不可行。

除非在 逻辑空间中 预先保证 每个集合 **并不相交**，而如果 $x \in A \;\wedge\; x \in B$，则 设定 $x_1 \in A$, $x_2 \in B$, $x_1 = x_2$, and x_1, x_2 are **path-connected**. 所以这需要用到 HoTT.

暂时 仍未知道 这种做法 是不是比「粗暴」法更有效率。

根据 paths 可以有如下的 topological「变形」：



(8.1.1)

但右边 仍是会出现 $x \in A$ accidentally 导致 $x \in B$ 的情况。

如果 neural network map F 的对象不是 ordered pairs $(x \in A)$, 可以是什么？A **proposition** can be regarded as a set of points x_i connected by paths.

8.2 HoTT levels

Voevodsky proposed these "levels":

$$\begin{array}{|c|c|} \hline \dots & \dots \\ \hline 2 & \text{2-groupoids} \\ \hline 1 & \text{groupoids} \\ \hline 0 & \text{sets} \\ \hline -1 & \text{(mere) propositions} \\ \hline -2 & \text{contractable spaces} \\ \hline \end{array} \tag{8.2.1}$$

On the proposition level, each type (space) either has a proof or is empty. Any two proofs on this level are considered **identical**. Thus, any 2 elements in a type on this level is connected by a homotopy (contractable space).

On the set level, the identity between 2 elements is a proposition $a \stackrel{?}{=} b$; it can be true or false. Thus a set can have multiple distinct elements.

On the next level, groupoids, the identity between any 2 elements is represented by a set.

Truncation

$||A||$ is a way to obtain the **truth value** of a type A, known as **truncation**. For Voevodsky, the truth value is always binary, the type of "mere propositions". I propose that it can be generalized such that, on HoTT level 0, it provides the **fuzzy** truth value $\in [0, 1]$.

8.3 What is homotopy?

8.4 Univalence axiom

在 HoTT 的 set 的层次，"=" 是一个 predicate, 根据我的理论可以看成是 fuzzy predicate.

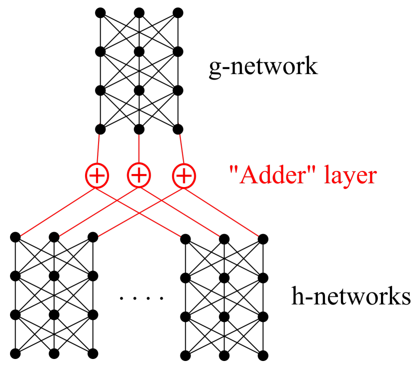
如果我没理解错误，univalence axiom 的意思 似乎是将 $=$ 的 fuzzy truth value「强制」成 binary.

9 Transfer to deep learning

这一节讨论 如何将上面提到的 logic structure 转移到 深度学习的 神经网络 (neural network, NN)。一般来说，很难将 额外的结构 impose 在 神经网络上，因为 NN 本身已经有很 “rigid” 的结构。在这方向上成功的例子是 CNN (convolutional NN)，其中 convolution $f * g$ 是由 “weight sharing” 模拟。但除此之外，一般很难在 NN 上添加结构。

Symmetric NN 是一种 在输入变量的 **置换**下 不变 (permutation invariant) 的神经网络。其解决 办法 是利用 以下的 函数形式：

$$f(x_1, ..., x_n) = g(h(x_1) + ... + h(x_n)) \tag{9.0.1}$$



(9.0.2)

注意在这个方案下，NN 是以 “black box” 作为 building blocks；内部结构不变。

范畴论 的好处是，将一切用 morphisms, compositions, pullbacks, adjunctions, 等 表示；这种做法 很容易用 NN implement.

9.1 Propositional aspect

在命题逻辑的层次上，我选择了最简单的特性，亦即是命题之间的 commutativity:

$$\begin{array}{ccc} A \wedge B & \equiv & B \wedge A \\ \text{下雨} \wedge \text{失恋} & \equiv & \text{失恋} \wedge \text{下雨} \end{array} \tag{9.1.1}$$

但我没有 fully exploit Heyting algebra or Boolean algebra 的结构，因为可以预见 将来是会 推广到 fuzzy-probabilistic logic，而后者的结构只需要 $\wedge$ 和 $\Rightarrow$，和 binary logic 有些不同（迟些解释....）

9.2 Predicate aspect

目前，一般的深度学习模型是比较 简单／粗暴 地作用在 自然语言句子的 **syntax** 层面，例如：

$$\text{“Je} \;\bullet\; \text{suis} \;\bullet\; \text{étudiant”} \stackrel{f}{\Longrightarrow} \text{“I} \;\bullet\; \text{am} \;\bullet\; \text{student”} \tag{9.2.1}$$

句子中的 words 是用 **Word2Vec** 方式 embed 到向量空间。但如果用了 Curry-Howard 对应，则会有些微不同，而这个微妙的差异 在数学上比较完美，实际上 computer implementation 会不会比较优胜？再看一次 Curry-Howard correspondence:

$$\begin{array}{c} A \Longrightarrow B \\ \text{- - - - - - - - - - - -} \\ \blacksquare \stackrel{f}{\longmapsto} \blacksquare \end{array} \tag{2.0.2}$$

这里 A 和 B 是**逻辑命题**，例如「我是学生」; f 将 A 命题里面的 witness ■ 映射到 B 命题里面。注意：「我是学生」这些**语法**上的信息，是以 f 的 **domain** 和 **co-domain** 表示的。

这一节 我们要讨论的是 命题及其内部 在计算机上的 implementation 问题。再看一次 图 (2.5.1) 的 两个层次:

$$\begin{array}{ccc} \overbrace{\text{Human (Socrates)}}^{A} & \Rightarrow & \overbrace{\text{Mortal (Socrates)}}^{B} \\ \downarrow & & \downarrow \\ \Omega & & \Omega \end{array} \tag{2.5.1}$$

红色 → 是 **predicates** 形成的层次。A 和 B 是两个不同的**空间**。在 A **内部**，Socrates 也是一个空间，Human 是另一个空间，而 Human(Socrates) 构成新的空间（透过 dependent type constructor Σ）。

假设 $X \in \mathcal{U}_1, P \in \mathcal{U}_2, P(X) \in \mathcal{U}_3$，（这些 $\mathcal{U}_i$ 是 type universes），在计算机上最简单的做法是令:

$$\mathcal{U}_3 = \mathcal{U}_1 \times \mathcal{U}_2 \tag{9.2.2}$$

这和 §2.5 说的 Σ type constructor 本质上是 Cartesian product 的说法吻合。

到此，我们发现，用 Curry-Howard 的做法 和「粗暴」的 syntactic 做法其实是一模一样的！有点失望，但我希望这些 漂亮的数学 不会是完全无用的....

9.3 Implementation of topology (points and sets)

是不是所有命题都 embed 到 命题空间 $\mathbb{P}$rop？还是有某种 working-space embedding？

这意思是说，从 long-term memory recall 进来，只要保证 inference makes sense 就行，constants 不必有长期的不变的位置。

或者可不可以说，是 constant 的 predicates 特性 决定其位置？

但 深度学习 似乎对 absolute positions 有优势。例如「嫲嫲」的位置。这是说，记忆中每个不同的客体 都有其 absolute position. 而另一种方法是 relative position：从记忆中 recall 来的物体，附带 很多 predicates，但其 position 是 on-demand 建构的。这两种做法有何分别？似乎关键是 recall mechanism；

但回忆必需是一个带有 context 的整体。问题是能不能做到 associative block recall.

这和 topology 有没有关？可不可以用 LTM 减少 F 的「负荷」？换句话说 F 只是负责 relative position 的 inference? 如果是 relative position 似乎有可能 reshape topology. 但仍是说不出 reshape topology 有什么好处，及其可能导致 errors 的问题.....

Reshape topology 带来 generalization 但也引入 errors.

问题是如果用了 topology 那么 F 会有哪些改变？命题是 $P(a) = 1$. 它的信息 似乎无论如何也是一个 tuple (P, a). 这会是一个 tuples to tuple 的 map，而这也就是 brutal syntactics. 但重点是：$P(a)$ 应该是一个**空间**。这空间的构成是透过 type constructor $\sum_a P(a)$. 这构成一个空间，而 F 是由 此空间到 彼空间 的一个映射。於是 F 并不是 tuples to tuple 的映射，而是 spaces to space 的映射。但这个映射是 witness 的映射，有点奇怪。换句话说，要用 记忆体 记住 witnesses，它们代表每一个命题。它们的位置就是命题的 syntax.

在空间中有某个 witness, 它的位置 正是 $P(a)$ 的 positional tuple，它被映射到新位置也是 $Q(a)$ 的 positional tuple，这岂不是和 tuple to tuple 一模一样?? 换句话说，无论怎样看，F 的映射 **必然是从 positional tuple to positional tuple**.

$P(a)$ 就是 (P, a)，syntactic mapping 是必然的。

那么 topology 又是什么? a 的位置是 a 作为 Atom 的位置，它本身可以看成是一个 predicate. 但，个别的 constants 有没有永久的 位置? 如果位置是永久的，则 recall 非常容易（直接 recall）。问题是要不要 reshape topology?

另一个问题是：开始时，一个 **新** constant 的位置是怎样 assign 的? 这显然和 **遗忘** 有关。一个简单的做法就是有 **constant 空间**，而这些 constants **位置是永久的**。

所以又回到问题：既然可以有 tuple-to-tuple, 那么 topological membership 要来做什么? 似乎就是用来 impose topological-metric **regularity** (ie, smoothness).

9.4 Modal aspect

10 Model-based AI

我提出的 BERT 的「逻辑化」AGI 模型 是基於 逻辑 syntax 的，换句话说是 用神经网络 模拟 $\vdash$，而这个 $\vdash$ 是 syntactic consequence. 现在考虑 $\vDash$，即 model-theoretic consequence.

最简单的 model 就是 point-set topology. 有如下对应：

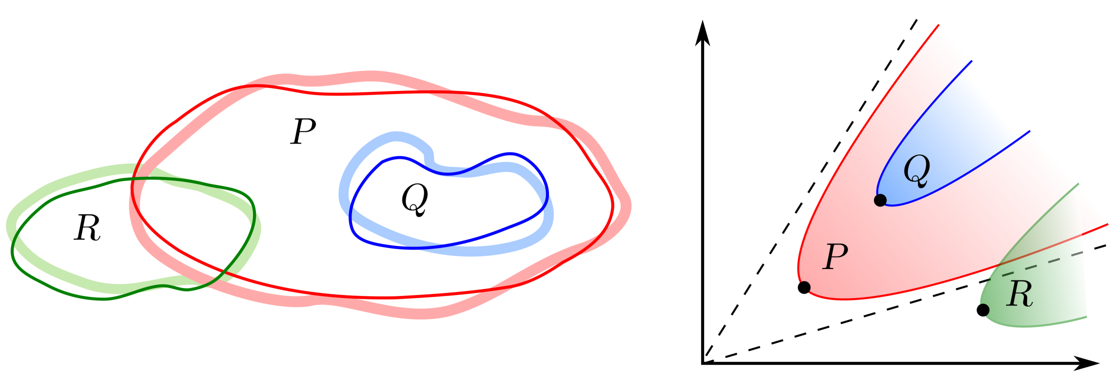
$$\begin{aligned} \text{propositions} &\;\Leftrightarrow\; \text{points} \in \text{regions or product of domains} \\ \forall\ \text{formulas} &\;\Leftrightarrow\; \text{sub-regions} \end{aligned} \tag{10.0.1}$$

Model-based (logic-based) AI 的其中一个主要问题是：在 syntax-based 情况下，我们可以在 working memory 引入一个命题 $P(a)$, 例如 $P =$「a 是不是质数」，而可以暂时不理会 $P(x)$ 在其他点上的真值；但在 model-based 情况下，model 包含了 P 的 extension（= 一个点集 或 区域），这表示我们立即知道每一个数是不是质数，这是不切实际的。这个问题或许可以称之为 "**omniscient predicates**" 的问题。

We need a quick way to instantiate a new object (point) but we don't want it to accidentally acquire some unwarranted properties. How to achieve this? Maybe maintain a discipline such that general predicates are located near the "top" position, they would not be contaminated with more specific predicates.

Then what is inference? What kind of changes would occur to the model?

Appearance of new propositions = appearance of new points / change of shape of regions.



(10.0.2)

Then how is the spatial model better than a syntactic representation (bunch of propositions)?

Definition 1. ***Deep learning*** *consists of a* ***hierarchical*** *structure with many levels (hence "deep") where the structure is learned to fit data. Such a learning algorithm must be* ***efficient****.*

第二个条件加上去是因为，「经典」逻辑 AI 的 学习算法 也符合「深度」条件，但它不够效率。

References

欢迎提问和讨论 ☺