

Introduction to strong AI

YKY (甄景贤 general.intelligence@Gmail.com)

previous collaborators and commenters:

Abram Demski

Ben Goertzel

Martin Magnusson

William Taysom

Russell Wallace

Pei Wang

© latest revision: October 8, 2018

Contents

0 导论	4
0.1 鸡与蛋问题, 罐头中的罐头刀	4
I 技巧	5
1 机器学习基础	6
1.1 归纳偏见与「没有免费午餐」	6
1.2 结构主义、后结构主义	6
2 逻辑	7
2.1 人脑思考的三大模式	8
2.1.1 Deduction	8
2.1.2 Abduction	8
2.1.3 Induction	8
2.2 命题逻辑	8
2.2.1 Boolean algebra	8
2.2.2 Boolean lattice	9
2.2.3 Knowledge and lattice structure	9
2.2.4 Heyting algebra	10
2.2.5 Boolean ring	10
2.2.6 Ideals and filters	10
2.2.7 人工智能中的 forward chaining	11
2.3 谓词逻辑 / 一阶逻辑	11
2.4 逻辑推理算法	11
2.4.1 消解算法 (resolution)	11
2.4.2 同一化算法 (unification)	11
2.5 二阶逻辑 / 高阶逻辑	11
2.6 λ -演算, 组合逻辑	11
2.7 各种逻辑结构之间的关系	11
2.8 Curry-Howard 同构	12
2.9 模型论	14
2.9.1 模型是怎样产生的?	14
2.9.2 与神经网络的关系	15
2.9.3 functorial semantics, “internal language”	15
2.10 Architecture of logic-based AI (LBAI) systems	16
2.11 代数逻辑、逻辑的几何化	16
2.11.1 Cylindric algebra	16
2.11.2 Cylindrical algebraic decomposition (CAD)	19
2.11.3 Quantifier elimination	19
2.12 范畴论、范畴逻辑	19
2.12.1 fibration	19
2.12.2 Lawvere 量词	20
2.12.3 Cut elimination	22
2.13 量子逻辑	22
2.14 项重写系统	22
2.15 图重写系统, 超图	22
3 不确定性	23
3.1 模糊性	23
3.2 机率	23

3.2.1 Bayesian networks	23
3.3 可信度	23
3.4 推理算法	23
3.4.1 MCMC (Markov chain Monte Carlo)	23
4 神经网络	24
4.1 神经科学	24
4.1.1 大脑结构	24
4.1.2 神经元	24
4.1.3 神经化学	24
4.2 神经网络的数学	24
4.2.1 非线性分析	24
4.2.2 拓扑度理论	24
4.2.3 同调论 / 奇点理论	24
4.2.4 调和分析	24
4.3 深度学习	24
5 进化算法	25
5.1 自然进化历史	25
5.2 进化算子及其算子谱	25
6 强化学习	26
6.1 控制论, 微分几何	26
6.2 最优化	26
II 功能组别	27
7 模式识别	28
7.1 视觉	28
8 信念修正, 真理维修	29
9 归纳学习	30
9.1 基於逻辑的归纳学习	30
10 自然语言	31
10.1 语法	31
10.2 语义	31
10.2.1 Abduction-as-interpretation	31
10.2.2 Montague 语法	31
10.2.3 Categorial 语法	31
11 计划	32
11.1 自动程式生成	32
III 系统	33
12 认知系统架构	34
13 记忆系统	35
13.1 工作记忆	35
13.2 事件记忆	35
14 实践	36
14.1 道德问题	36

14.2 商业化	36
专有名词与缩写	37
参考文献	38
索引	38
致谢	38

0 导论

0.1 鸡与蛋问题，罐头中的罐头刀	4
-------------------	---

0.1 鸡与蛋问题，罐头中的罐头刀

Part I

技巧

1 机器学习基础

1.1 归纳偏见与「没有免费午餐」	6
1.2 结构主义、后结构主义	6

1.1 归纳偏见与「没有免费午餐」

1.2 结构主义、后结构主义

2 逻辑

“The only way to rectify our reasonings is to make them as tangible as those of the Mathematicians, so that we can find our error at a glance, and when there are disputes among persons, we can simply say: Let us calculate [calculemus], without further ado, to see who is right.”

— Leibniz

2.1 人脑思考的三大模式	8
2.1.1 Deduction	8
2.1.2 Abduction	8
2.1.3 Induction	8
2.2 命题逻辑	8
2.2.1 Boolean algebra	8
2.2.2 Boolean lattice	9
2.2.3 Knowledge and lattice structure	9
2.2.4 Heyting algebra	10
2.2.5 Boolean ring	10
2.2.6 Ideals and filters	10
2.2.7 人工智能中的 forward chaining	11
2.3 谓词逻辑 / 一阶逻辑	11
2.4 逻辑推理算法	11
2.4.1 消解算法 (resolution)	11
2.4.2 同一化算法 (unification)	11
2.5 二阶逻辑 / 高阶逻辑	11
2.6 λ-演算, 组合逻辑	11
2.7 各种逻辑结构之间的关系	11
2.8 Curry-Howard 同构	12
2.9 模型论	14
2.9.1 模型是怎样产生的?	14
2.9.2 与神经网络的关系	15
2.9.3 functorial semantics, “internal language”	15
2.10 Architecture of logic-based AI (LBAI) systems	16
2.11 代数逻辑、逻辑的几何化	16
2.11.1 Cylindric algebra	16
2.11.2 Cylindrical algebraic decomposition (CAD)	19
2.11.3 Quantifier elimination	19
2.12 范畴论、范畴逻辑	19
2.12.1 fibration	19
2.12.2 Lawvere 量词	20
2.12.3 Cut elimination	22
2.13 量子逻辑	22
2.14 项重写系统	22
2.15 图重写系统, 超图	22

2.1 人脑思考的三大模式

2.1.1 Deduction

2.1.2 Abduction

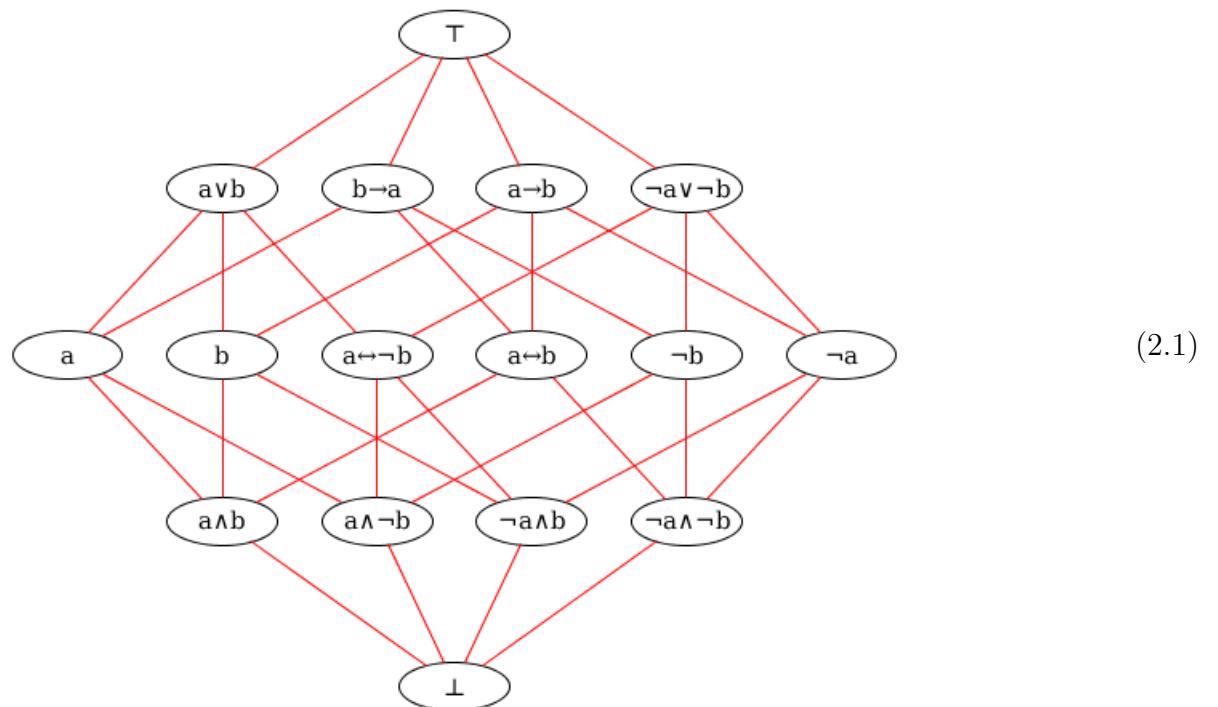
2.1.3 Induction

2.2 命题逻辑

2.2.1 Boolean algebra

假设大家都熟悉 Boolean algebra，一个比较抽象的定义是：a Boolean algebra is a **complemented distributive lattice**.

The Boolean algebra with 2^4 elements:



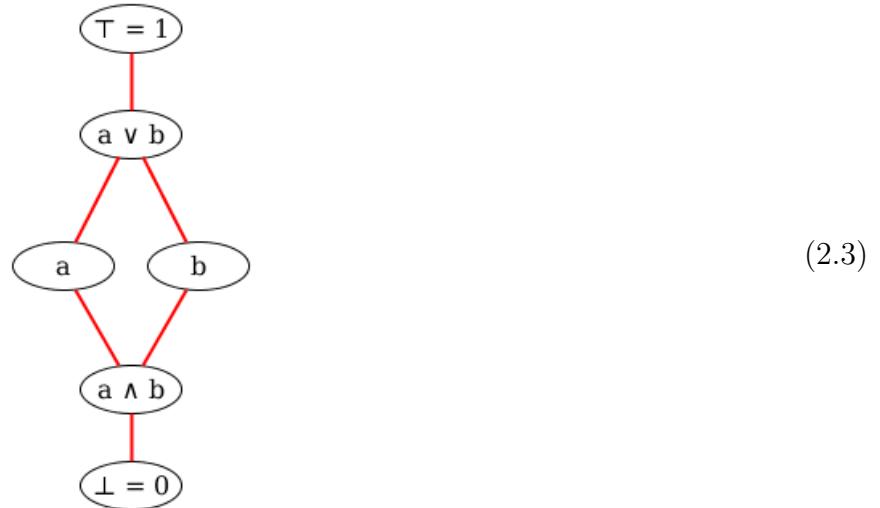
其中 $a \rightarrow b \equiv \neg a \vee b$, $b \rightarrow a \equiv a \vee \neg b$ 。

2.2.2 Boolean lattice

$$\begin{aligned}
 a \rightarrow b = \top &\Leftrightarrow \neg a \vee b = \top \\
 &\Leftrightarrow a \vee b = a \\
 &\Leftrightarrow a \leq b \quad (\text{by definition of lattice})
 \end{aligned} \tag{2.2}$$

所以 $a \leq b$ 定义了一个 order relation, 这就是 the Boolean lattice associated with the Boolean algebra, 两者是等价的。

这是包含两个命题的 Boolean lattice (但似乎不是 complete atomic Boolean lattice):



對於 任意 a 恒有:

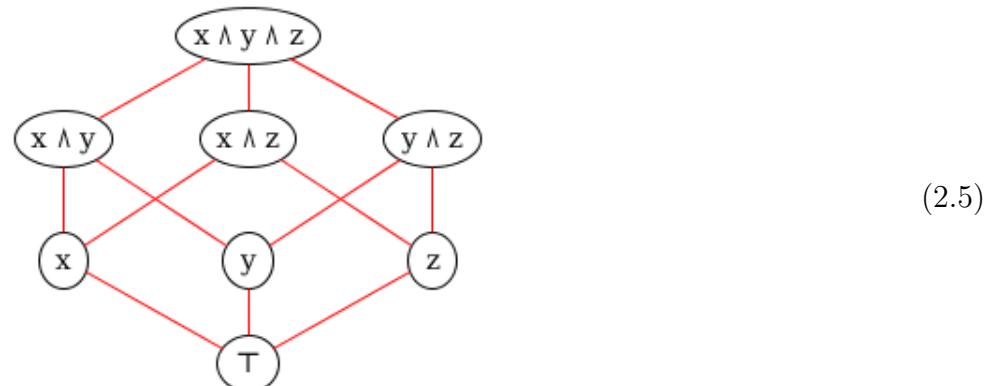
$$\begin{aligned}
 \perp \leq a &\leq \top \\
 \perp \rightarrow a &\rightarrow \top
 \end{aligned} \tag{2.4}$$

因为: 由假的前提 (\perp) 可以推出任意结论 (“*ex falso sequitur quodlibet*”),
而由任意前提都可以推出 恒真命题 (\top)。

$a \rightarrow b \Leftrightarrow a \leq b$ 但方向相反, 原因是 $0 \leq 1$, 在 lattice 中越往下的命题 越难满足。

2.2.3 Knowledge and lattice structure

如何用 lattice 表达 AI 中的逻辑结构? 似乎有两个方法。首先, 将所有可能出现的命题集合都表达成 lattice 中的某些节点:



换句话说, lattice 中每个节点 表示 mental state 的一个 状态。进行逻辑思考时, 由一个状态跳到另一个状态, 这 transition function 就是 F (亦即我提议用 神经网络 模拟的函数)。换句话说, lattice 代表 状态空

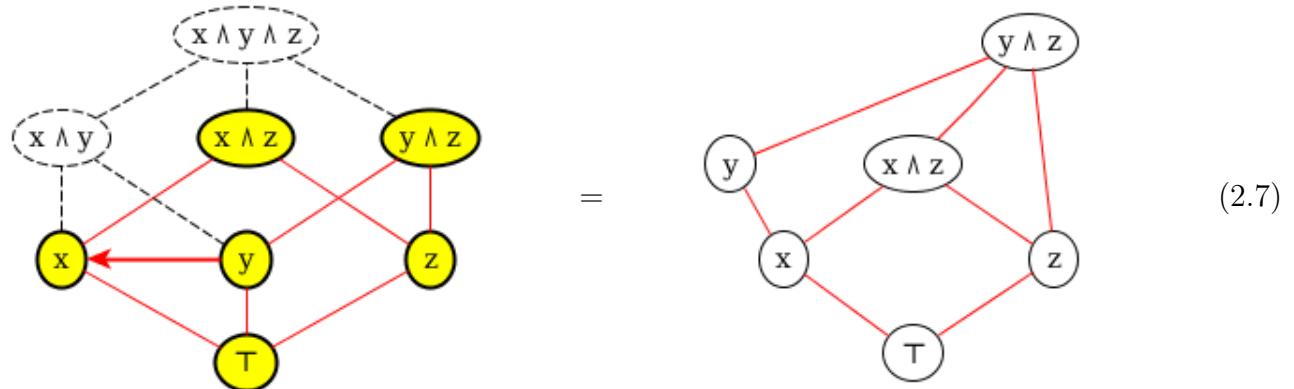
间，但 知识 是储存在 F 里面。所谓 知识 其实就是这样的一些 逻辑 rules (或者叫 conditionals):

$$p \rightarrow q \quad (2.6)$$

例如 (吃了不洁食物) \rightarrow (肚痛)

在上例中 (fig. 2.5)，状态空间 包含所有可能的命题组合，可以说这个 lattice 是 “regular” 的 (但这 terminology 可能不正确)。

如果在 lattice 中加入一些已知的 knowledge, 例如 $y \rightarrow x$, 则某些 节点 会消失, 而且加入了一个新的 edge:



这时, 状态 lattice 变得 “irregular”, 但 F 只需在这 lattice 中容许的节点上移动。

2.2.4 Heyting algebra



2.2.5 Boolean ring

给定一个 Boolean algebra, 可以定义:

symmetric difference

$$\begin{aligned} a + b &= (a \wedge \neg b) \vee (\neg a \wedge b) \\ a \cdot b &= a \wedge b \end{aligned} \quad (2.8)$$

则 $(A, +, \cdot)$ 构成一个 Boolean ring。

$$a + a = 0 \quad (2.9)$$

所以 Boolean ring 的 characteristic 是 2。

在 Boolean ring 里, complement 可以这样表示:

$$\neg a = a + 1 \quad (2.10)$$

(在数学上, 据说 ring 较方便研究, 但我暂时不知其细节。)

2.2.6 Ideals and filters

Ideal 的概念来自 algebraic geometry。如果我们有一组代数方程:

$$\begin{aligned} f_1 &= 0, \\ &\vdots \\ f_s &= 0 \end{aligned} \quad (2.11)$$

则由 f_s 组成的 线性组合 也会是这组方程的解:

$$h_1 f_1 + h_2 f_2 + \dots + h_s f_s = 0. \quad (2.12)$$

形如左边那种元素，就是这 ideal 的元素。The ideal generated by f_s , 记作 $\langle f_1, \dots, f_s \rangle$, 它可以看成是方程组 $f_1 = f_2 = \dots = f_s = 0$ 的「多项式后果」(polynomial consequences) 的全体集合。[Cox, Little, and O'Shea 2007] p.31.

在逻辑上，ideal 对应於一些逻辑式子的 consequences, 但由於在 代数几何 中，ideal 的定义是 algebraic equations = 0, 但逻辑中 真 = 1, 所以需要 ideal 的 dual concept, 即 filter。

在 Boolean algebra 中，如果问「某一命题 q 是不是前提 Γ 的结论？」其答案等於问： q 是不是屬於由 Γ 生成的 ideal 之中？在 computational algebraic geometry 中，ideal membership 的问题可以用 Gröbner basis 的方法解答。暂时我不清楚这个方向有没有优势。

我觉得现时一个重要问题是：如果我们用神经网络 F 近似地 模拟 逻辑 \vdash ，而逻辑有其代数结构，则这代数结构如何影响 F 的结构，及 F 的 learning algorithm？这问题会在 §2.9.2 考虑。

2.2.7 人工智能中的 forward chaining

2.3 谓词逻辑 / 一阶逻辑

2.4 逻辑推理算法

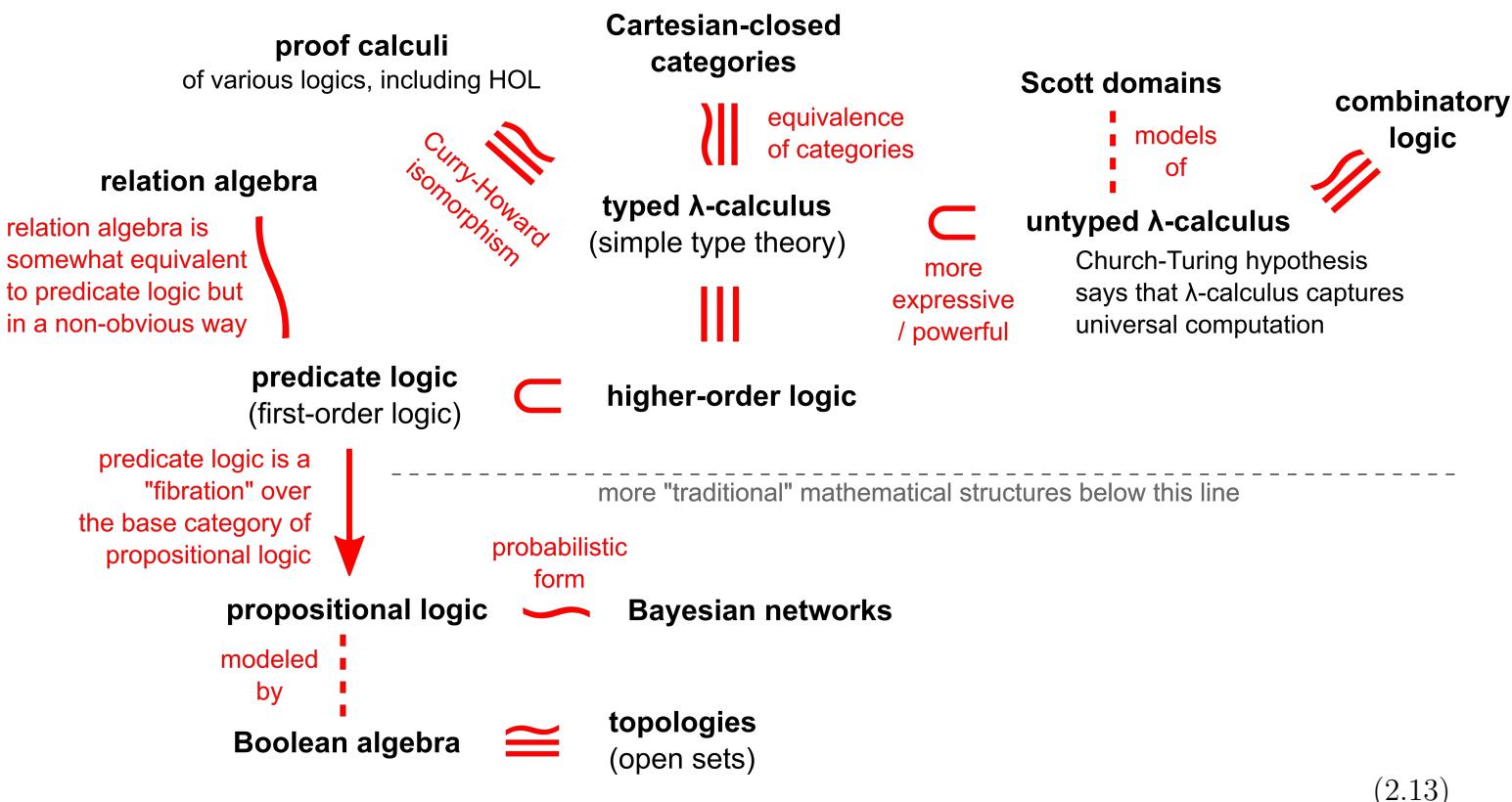
2.4.1 消解算法 (resolution)

2.4.2 同一化算法 (unification)

2.5 二阶逻辑 / 高阶逻辑

2.6 λ -演算，组合逻辑

2.7 各种逻辑结构之间的关系



在上图中最「麻烦」的似乎是 由 propositional logic 提升到 predicate logic 的 **fibration**。从命题逻辑上升到 type theory 可以透过 Curry-Howard isomorphism (下节介绍)；在这过程中，propositions 对应 **types**, predicates 对应 **dependent types**。

举例来说：

命题	『约翰是男人』	$Male(john)$	
命题	『约翰爱玛莉』	$Loves(john, mary)$	(2.14)

Predicate 是一种函数，它吃一些 objects，吐出一个 proposition。

在 type theory 那边，dependent type 吃一些 terms，吐出一个 type。例如 $ARRAY(n)$ 是一个 dependent type，它吃掉一个整数 n ，吐出一个长度为 n 的 $ARRAY$ type。

两边之间有这样的 Curry-Howard 对应：

Logic	Type theory	
formulas	\rightsquigarrow types	
proofs	\rightsquigarrow terms	
predicates	\rightsquigarrow dependent type: terms \rightarrow type	(2.15)

Dependent type 是一个吃 terms 吐出 type 的函数。它对应於逻辑 predicate，换句话说，predicate 是一个吃 proofs 吐出 proposition 的东西。这有点奇怪，因为 predicate 通常吃的是逻辑里的 objects (也可以叫做 logic constants)。但细想一下，如果 $john$ 存在於 $Male$ 这个集合内，则 $john$ 顺理成章地可以看成是 $Male(john)$ 此一命题的 proof。换句话说，逻辑 objects 可以看成是逻辑上的 proofs。

Simple type theory 与 CCC 有 equivalence of categories (在 [Lambek and Scott 1986] 书中有论述)；在这等价关系中，types 对应於 morphisms (例如 morphism $f : A \rightarrow B$ 对应於 $x : A \vdash f(x) : B$)。换句话说，似乎要模拟 predicate logic 的关键是需要某些 “morphisms”。

注意：simple type theory 亦被叫作 higher-order logic，它是 first-order logic 的扩充，但这扩充并不需要经过 Curry-Howard isomorphism，因此往往造成混淆。现代逻辑的研究似乎特别关注 Curry-Howard 同构。

2.8 Curry-Howard 同构

λ -calculus 是一切「计算」(computable functions) 的形式。Type theory 的作用是 enforce type discipline 以避免一些 paradox，例如 untyped λ -calculus 里面有 Curry paradox。

在 proof theory 里面，逻辑的 **后设法则** (meta-logical rules) 描述怎样由一些 逻辑式子 推导 到另一些 式子。由 Brouwer-Heyting-Kolmogorov 提出的想法是：将 逻辑蕴涵 (implication) 看成是一些 functions 或 mappings。这个想法再经 Curry-Howard 进一步发展，认为 proofs 就是「计算」，所以对应於 λ -terms。

Type theory	Logic	
(“programs”) λ -terms	\sim proofs	
	types \sim propositions	(2.16)

详细可参看 [Sørensen and Urzyczyn 2006] 一书。

同一书中，还有以下两个列表：

Type theory	Logic
type	\sim formula
type variable	\sim propositional variable
type constructor	\sim connective
function space	\sim implication
product	\sim conjunction
disjoint sum	\sim disjunction
empty type	\sim absurdity
term	\sim proof
object variable	\sim assumption
constructor	\sim introduction
destructor	\sim elimination
redex	\sim proof detour
reduction	\sim normalization
normal form	\sim normal proof
inhabitation	\sim provability

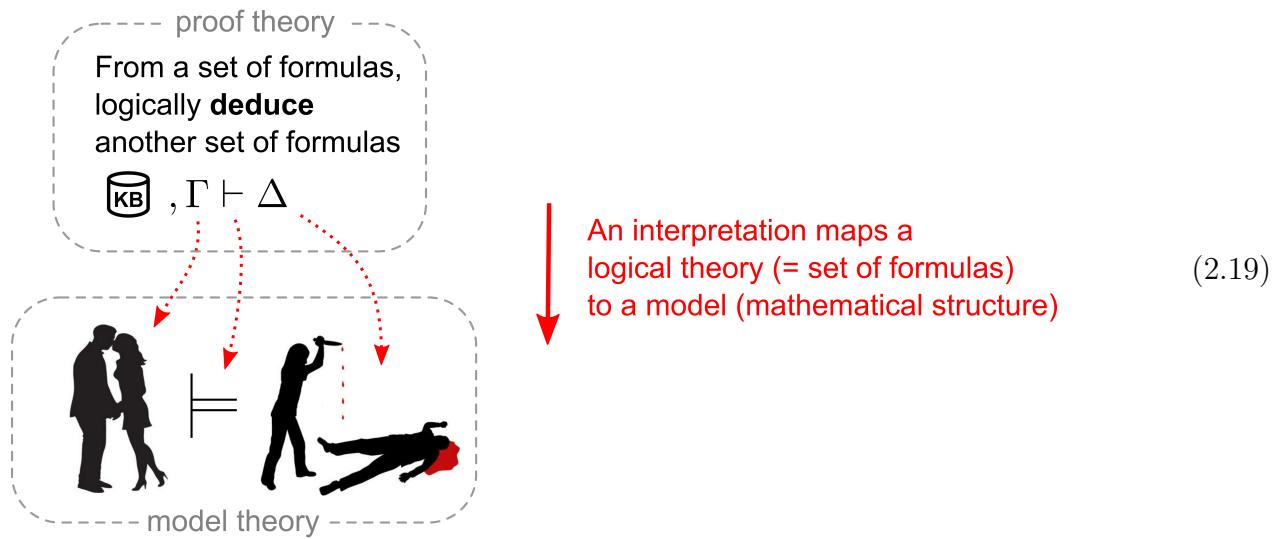
(2.17)

Type theory	First-order logic
types	\sim formulas
terms	\sim proofs
dependent types	\sim connectives, eg. \wedge, \vee
dependent type	\sim atomic formula
type constructors	\sim predicates / relations
product type	\sim universal formula
abstraction $\lambda a.M$	\sim proof by generalization
application Mt	\sim proof by specialization
abstract type or module	\sim existential type
encapsulation	\sim proof by \exists -introduction
opening a package	\sim proof by \exists -elimination

(2.18)

2.9 模型论

模型论由 Tarski 开始研究，它的目的是找出一些数学结构（通常是代数结构），这些结构符合某个 logic theory 的描述。Logic theory 即一些逻辑式子的集合。



(逻辑是指丈夫有婚外情，推导出妻子因妒忌谋杀丈夫)

我理解 model 的意思，大概像「脑海中的小电影」。例如，当我提及「妻子在家斩杀丈夫」这句子时，读者在脑中 re-construct 这个场景。这时如果有 query 问：「妻子当时是不是在酒吧喝酒？」或者「现场是不是会遗下大量血迹？」人脑’ul 可以从 models 中直接 read off 这些答案。

广义上来说，model 就是一个和 “the real thing” 有某种程度相似的结构，例如尺码缩小了或变形了。而 logic theory 是一堆命题，它表面上不像 real thing，但从广义来说，它也可以叫 model。

2.9.1 模型是怎样产生的？

在认知科学里，「模型」这概念并没有数学上严谨的定义，所以我提出一个观点：所有模型都是由 pattern recognition 的逆运算产生的：

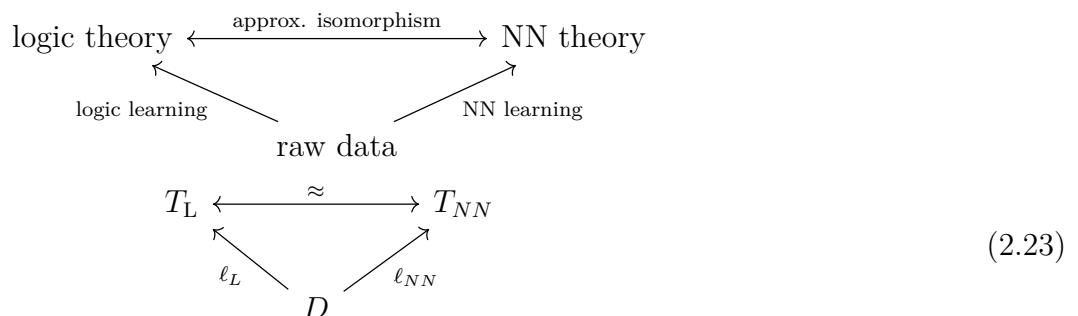
$$\boxed{\text{raw data}} \xrightleftharpoons[\text{model instantiation}]{\text{inductive learning}} \boxed{\text{theory}} \quad (2.20)$$

Pattern recognition 也可以叫 inductive learning。Models 存在於 raw data 的空间中，这是合理的，因为 models 其实就是一些「被反向想像出来的 data」。各种机器学习的算法，产生结构和性质不同的 theories，而这些 theories 逆向产生的 models 也会有不同的性质：

$$\boxed{\text{raw data}} \xrightarrow[\ell_L^{-1}]{\ell_L} \boxed{T_L \text{ [logic theory]}} \quad (2.21)$$

$$\boxed{\text{raw data}} \xrightarrow[\ell_{NN}^{-1}]{\ell_{NN}} \boxed{T_{NN} \text{ [NN theory]}} \quad (2.22)$$

可以想像 logic theories 和 neural theories 之间有某种近似的等效 \approx ：



我们的目的是找出这个 \approx 可资利用的优势。

早在 2000 年, Jocelyn [Ireson-Paine 2000] 已提出了 在机器学习中 generalization 可以理解成 adjunction, 这看法和我的理论是一致的。

2.9.2 与神经网络的关系

主导思想: 用神经网络 F 模拟逻辑的 \vdash 。

深度神经网络 可以自动学习出 知识表述 (representations), 这 representation 的 涌现 (emergence) 是由 目标函数 的最优化「迫」出来的。

一个重要的问题是: 逻辑的代数结构如何影响 F 的结构, 及 F 的 learning algorithm?

\vdash 似乎是一个 multi-valued function, 但 F 会是 single-valued。在这个 选择 过程中, 似乎蕴含了 proof search (更正确地说是 consequence search, 实际 implement 时可以用 best-first search, A*-search 等)。

逻辑的代数结构似乎就是 lattice 结构 (如果不考虑 predicate logic 的 fibration), 所以 consequence search 是在一个 lattice 中进行。

其实共有 3 个层次:

- **Propositional lattice:** 这是 deduction 的 search space。
- **KB lattice:** 每个 (propositional) lattice 是由 $x \rightarrow y$ 这样的逻辑 rules 决定, 换句话说是 KB 的内容。换句话说, 每个 KB 决定一个 lattice。但, 不同的 KB 也组成另一个更大的 lattice。这就是 inductive learning 里面的 hypothesis space。
- 而, 不同的 logics 也可以组成一个「更大更大」的 lattice (in the sense of universal algebra), 但我觉得这不是重点, 因为在 logic 的方向上很容易就已经达到 Turing universal 的程度, 所以我们暂时不必深究用哪个 logic。这是 meta-logical learning 的 search space。

F 的迴路构成一个 RNN (recurrent neural network)。RNN 的特性是它会自动地 iteratively settle down to a certain interpretation, which seems to be an advantage. 每次 iteration 是在寻找 global minimum, 换句话说, iteration 期间 是尝试一些 combinations of possible solutions, analogous to proof search in LBAI.

总结一下以上的论述:

- Logic's algebraic structure gives rise to lattice structure, which is the structure of the search space of deduction.
- The set of all KB's form a lattice which is the hypothesis space of inductive learning. This lattice structure is induced by the lattice structure of logic. The search for optimal F occurs in this lattice.

2.9.3 functorial semantics, “internal language”

将 模型论 用 范畴论 的方法表述, 得到 functorial semantics。

在 [Crole 1993] 一书里有解释:

Internal language 是指 一个范畴 \mathbf{C} 诱导出的 algebraic theory $Th(\mathbf{C})$, 叫作 \mathbf{C} 的 internal language. This language is extremely useful because it allows us to reason about the category \mathbf{C} as though it were the category **Set** of sets and functions. For example, if $f : A \rightarrow B$ is a morphism of \mathbf{C} , then there is a **proved term** $x : A \vdash f(x) : B$ in $Th(\mathbf{C})$.



(还要解释一下, 这 internal language 似乎是 Curry-Howard isomorphism 的一种形式....)

2.10 Architecture of logic-based AI (LBAI) systems

经典 LBAI 的基本运作是这样的（即 经由 KB 的逻辑法则，从 current state 推导到下一个结论）：



KB 是一些 条件命题 (logic rules) 的集合。

逻辑推导 分解成几个动作 完成：

- unification: 即 pattern matching, 寻找 KB 中 可用的 条件命题
- resolution: 应用条件命题 推导出新的结论

问题是，经典 LBAI 中的 \vdash 是 “symbolic” 的。换句话说：经典 LBAI 最大的问题是，它处理的元素之间没有 semantic distance，而只有 syntactic distance，而后者基本上是没有用处的。为了解决这问题，似乎需要使用 model theory，因为 models 的内部可以用 semantic distance 量度。

在 认知科学 里，關於 pattern recognition 的问题，一直有 model-based view 和 theory-based view 等论点的争议。究竟人的认知 是基於：

- logic theory, 即一些 逻辑法则 的集合
- prototypes / exemplars, 记忆一堆例子，然后新的例子 用 similarity metric 量度
- models ?

例如 人 怎样辨认「水果」这概念？例如「tomato」不是甜的，所以属於 borderline 水果，这似乎是根据 逻辑法则 决定的。

在 经典 LBAI 里，如果要回答「妻子在案发当时是不是在酒吧喝酒？」需要用到一些逻辑法则，例如：「如果 A 用刀斩 B，则 A 需要在 B 的近距离」，这样可以推导出「妻子不可能在酒吧」。但问题是，我们假设知识库 KB 中有这样的一些法则 似乎很牵强。LBAI 支持者通常的回答是：知识库中有成千上万的逻辑法则，多不胜数，这只是其中一条。但细想之下这个做法似乎有点不切实际。

另方面，如果 我 凝视面前自己的手掌，我脑中 建立的 model 是类似一个 立体的 block 加上几个 cylinders。其实这个 model 也可以表述为一组 逻辑命题，所以 model 和 logic theory 之间没有明确分野。甚至一个 神经网络 的 neuron，它只负责辨认一个微小的 feature，但这个 feature 也可以转变为 逻辑命题。

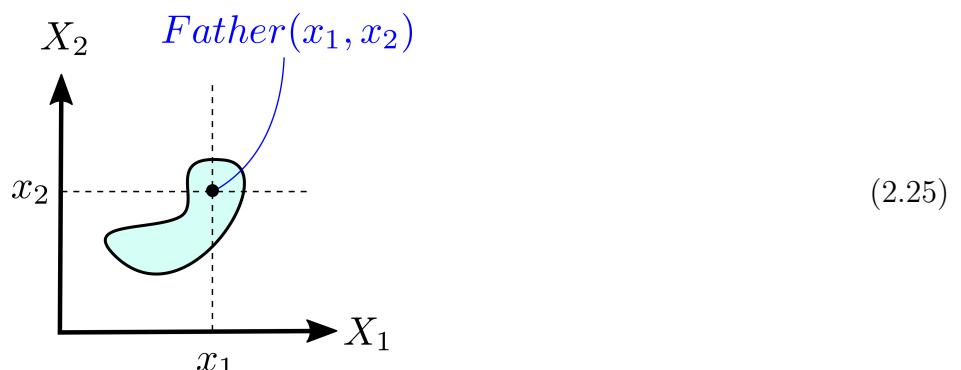
2.11 代数逻辑、逻辑的几何化

谓词逻辑 的 代数化 (algebraization) 有两个做法：Alfred Tarski 的 cylindric algebra 和 Paul Halmos 的 polyadic algebra。如果不用 谓词逻辑，也可以有 relation algebra 这种代数。

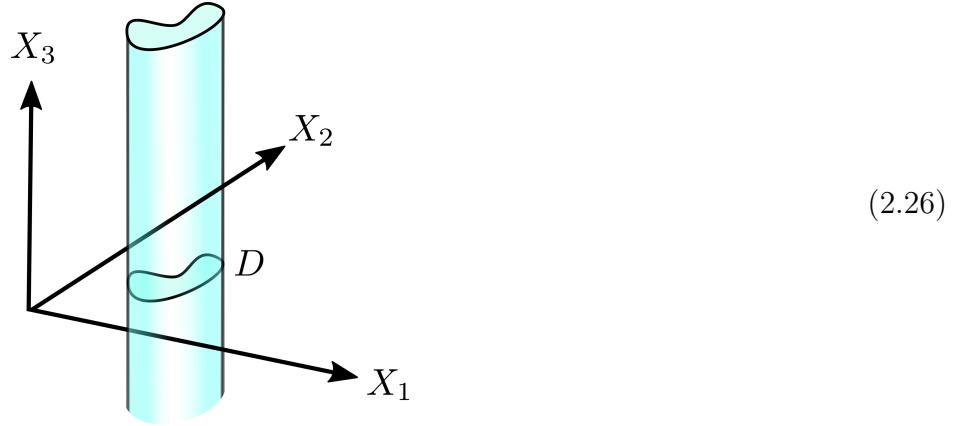
2.11.1 Cylindric algebra

Cylindric algebra 由 Tarski 发明，目的是给出 first-order logic 的一种 代数形式 (algebraization)。

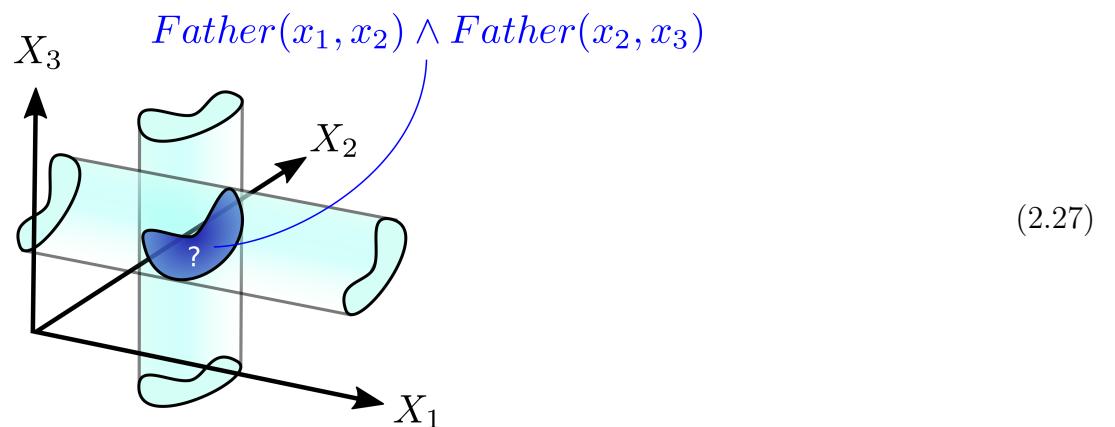
例如，一个关系 $x_1R x_2$ ，也可以记为 $R(x_1, x_2)$ ，它存在於 $X_1 \times X_2$ 这两个 domain 的 Cartesian 乘积之中：



Topologically, cylinders 的产生是来自 domains 的 Cartesian 乘积,
例如 $\text{Father}(x_1, x_2)$ 是 $X_1 \times X_2$ 内的一个区域 D , 但 x_3 是“don't care”, 所以「乘以」domain X_3 的全体, 亦即 $D \times X_3$, 因而产生 cylinder:



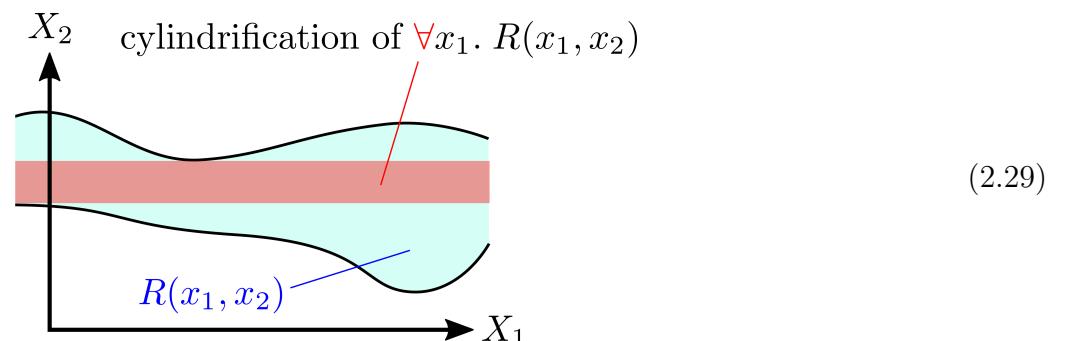
两个关系的 composition $R_1 \circ R_2$ 是他们的 cylinders 的 intersection:



这个 intersection 的形状可以很复杂, 例如当两个 L 形的 cylinders intersect 时, 会产生这个体积:



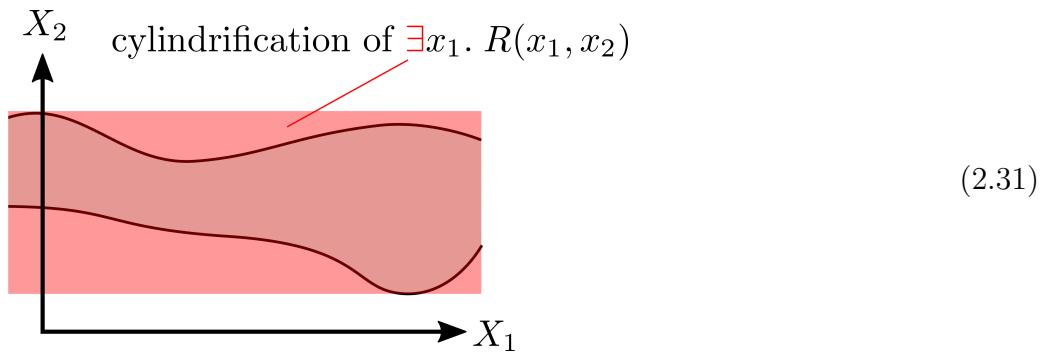
量词 \forall 的 cylindrification 如下图所示:



在上例中, \forall 的 cylindrification 是这个集合:

$$\blacksquare_{\forall} = \{(x_1, x_2) \mid \forall \hat{x}_1. R(\hat{x}_1, x_2)\} \quad (2.30)$$

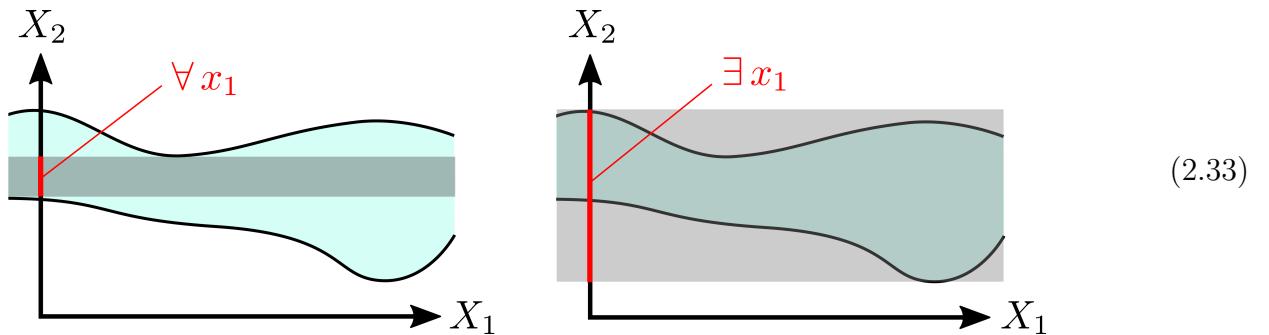
同样地,



\exists 的 cylindrification 是这个集合:

$$\boxed{\exists} = \{(x_1, x_2) \mid \exists \hat{x}_1. R(\hat{x}_1, x_2)\} \quad (2.32)$$

或者 等效地, 可以将 \forall 和 \exists 表示成 定义域内 X_2 成分的 投影:



这时, 我们定义 投影 函数:

$$\pi_1(x_1, x_2) = x_1, \quad \pi_2(x_1, x_2) = x_2 \quad (2.34)$$

则上面的两个 投影后的集合 可以表示成:

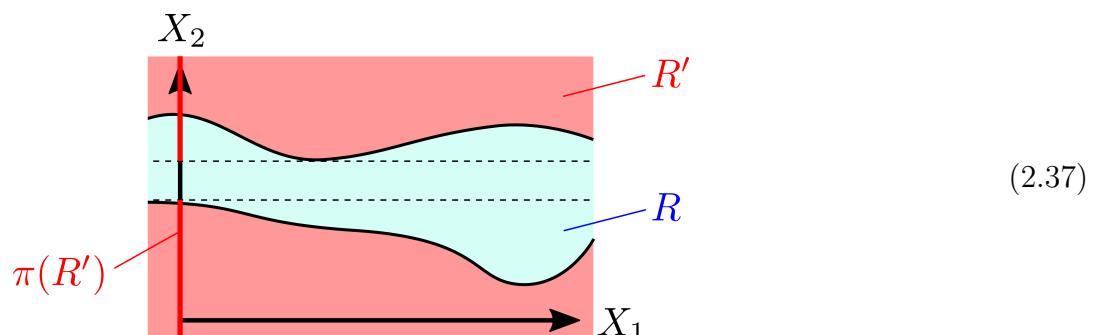
$$\begin{aligned} \boxed{\exists} &= \exists_{x_1} = \{x_2 \mid \exists \vec{x}. [x_2 = \pi_2(\vec{x}) \wedge \vec{x} \in R]\} \\ \boxed{\forall} &= \forall_{x_1} = \{x_2 \mid \forall \vec{x}. [x_2 = \pi_2(\vec{x}) \Rightarrow \vec{x} \in R]\} \end{aligned} \quad (2.35)$$

注意: 在第二式中的 \Rightarrow 是必需的, 不能用 \wedge 代替, 这是很微妙的区别, 而两式并不完全对称。
被量词 $\exists x$ 和 $\forall x$ 束缚 (bound) 的变量, 是那个被投映忽略了的变量, 而不是被投映到的变量。

上面两式可以用更简洁的方法描述:

$$\begin{aligned} \exists_{x_1} &= \pi_2(R) \\ \forall_{x_1} &= \pi_2(R')' \end{aligned} \quad (2.36)$$

换句话说: \exists 是 R 经过投映 π_2 的 象 (image),
 \forall 是 R 的 补集 (complement) 的投映的补集 (如下图):



F William Lawvere 在 1960's 年代提出了基於 范畴论 的更深入的想法, 见 §2.12。

2.11.2 Cylindrical algebraic decomposition (CAD)

CAD 是在 real algebraic geometry 中很重要的一个 algorithm, 或许和 cylindric algebra 有些关系。



2.11.3 Quantifier elimination

2.12 范畴论、范畴逻辑

范畴逻辑的要点可概括为:

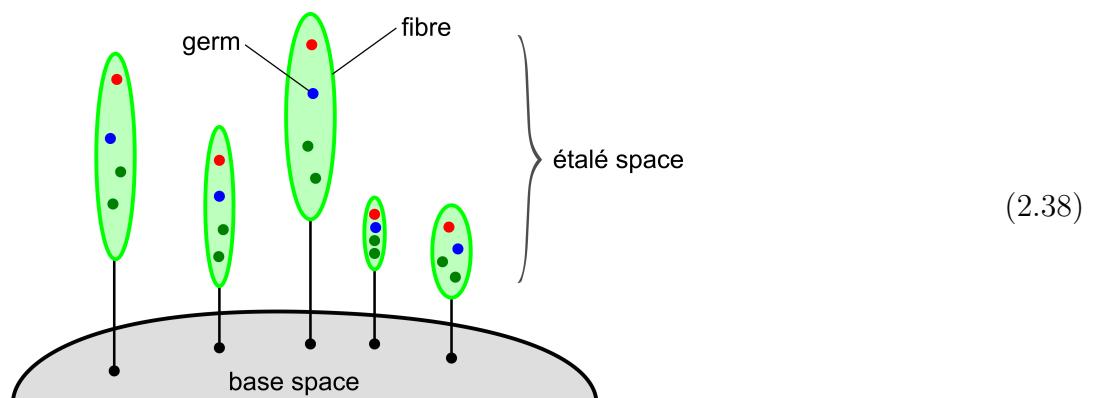
- predicate logic is a **fibration** over the base of propositional logic
- **quantifications** \exists and \forall are **adjoints** to the substitution functor
- the **equality** predicate $=$ is **adjoint** to the substitution functor
- set **comprehension** is **adjoint** to the truth functor

这部分的 textbook 包括: [Jacobs 1999]

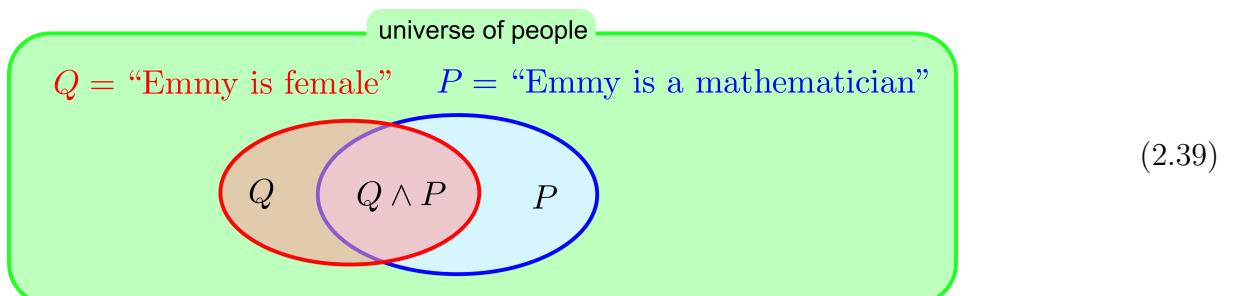
由於 adjunctions 在数学中是 无处不在 的, 这些 adjoint 关系可以方便我们找出 逻辑结构 在 vector space 或 metric space 上的 嵌入, 从而应用到 深度学习 中。

2.12.1 fibration

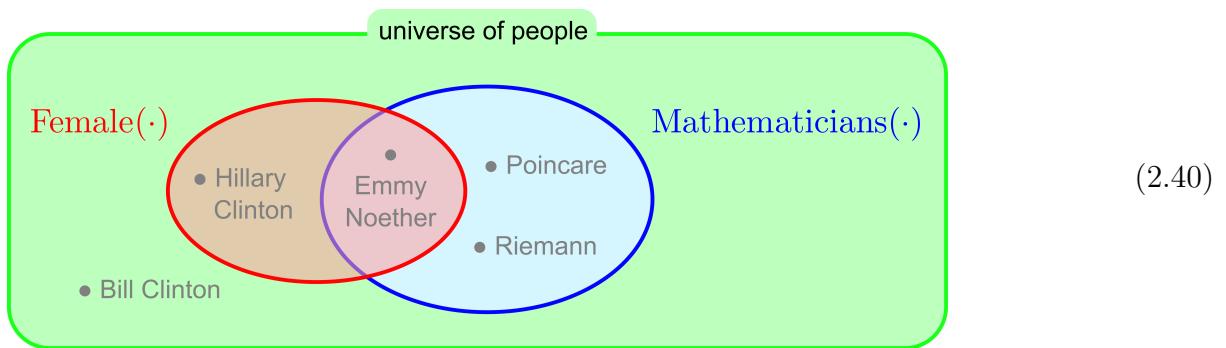
Fibration 的定义:



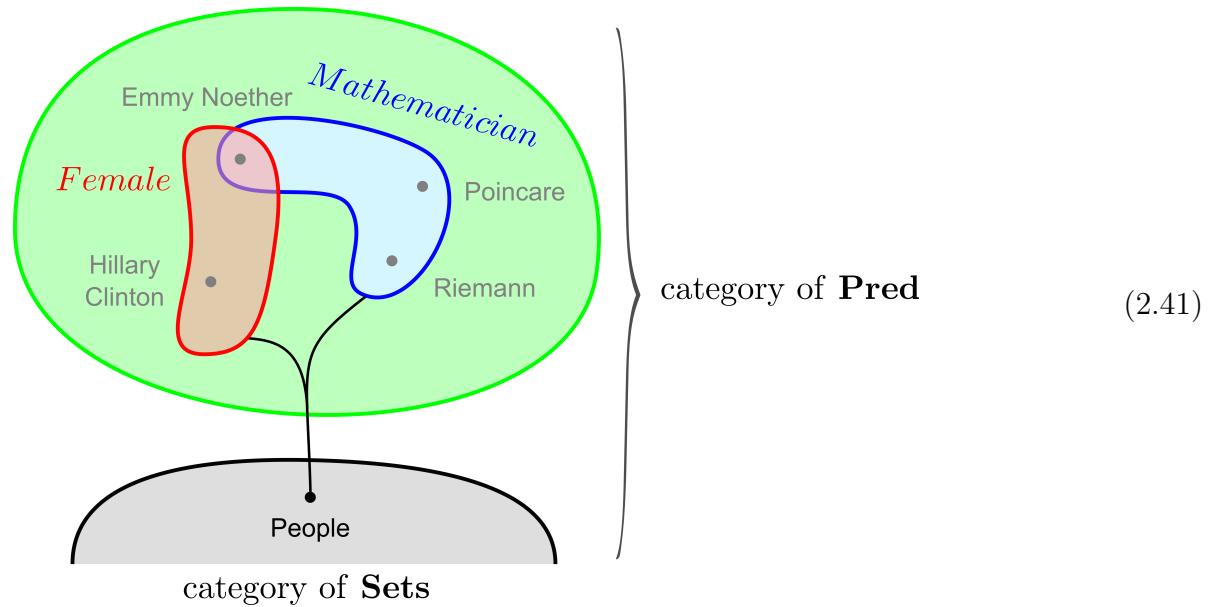
命题逻辑 是一种 拓扑 (开集) 结构:



如果命题的 内部 有结构，则变成 谓词逻辑：



谓词逻辑 可以 看作是 命题逻辑 之上的一个 **fibration**, 记作 $\downarrow_{\text{Set}}^{\text{Pred}}$:



2.12.2 Lawvere 量词

如果要将 逻辑 表述成 范畴论, 则所有 逻辑运算 都应该表示成 某些 mappings。其中最难搞的是 \forall 和 \exists , 因为表面看上去它们是一些 meta-logical 符号而已。我觉得 Lawvere 提出的解决办法是很惊人的。

[Lawvere 1963] 提出, \exists 和 \forall 分别是某个 substitution map f 的左和右 **adjoint functors**:

$$\exists \dashv f \dashv \forall \quad (2.42)$$

f 是一个叫 “weakening” 的函子, 它的作用是将 论域 X 扩充 映射到 $X \times X_0$, X_0 是一个新的 variable:

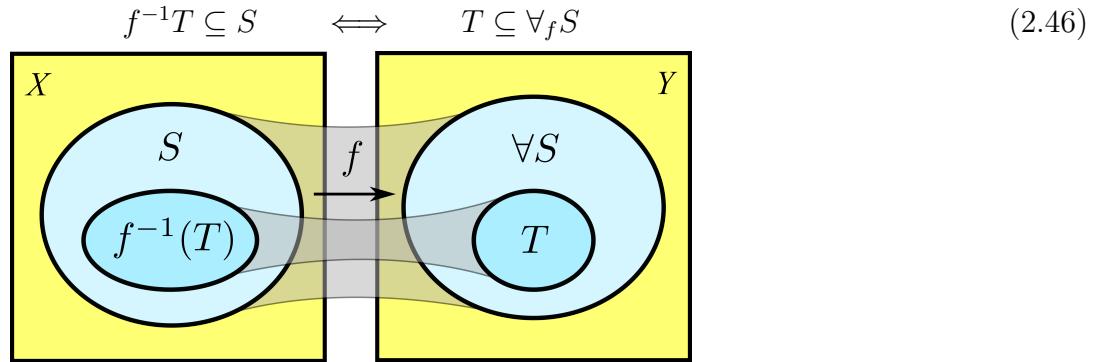
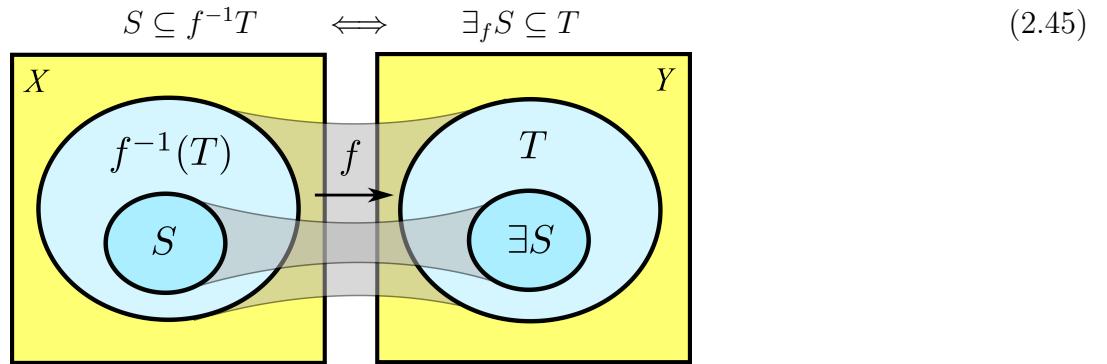
$$f : X \rightarrow X \times X_0 \quad (2.43)$$

假设 X 和 Y 是任意的 论域, 而 $f : X \rightarrow Y$ 代表 论域的 转变, 所以 f 是一种 “substitution map”。则可以定义:

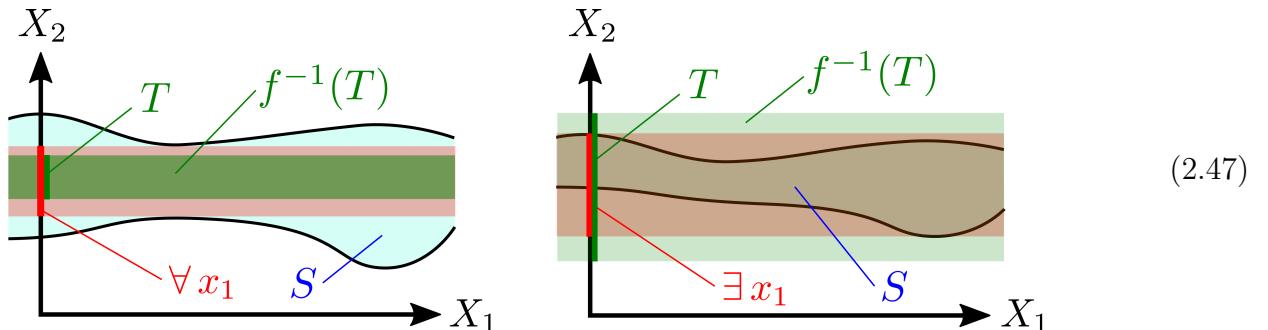
$$\begin{aligned} \exists_f &= \{y \in Y \mid \exists x \in X. [y = f(x) \wedge x \in S]\} \\ \forall_f &= \{y \in Y \mid \forall x \in X. [y = f(x) \Rightarrow x \in S]\} \end{aligned} \quad (2.44)$$

以上对 \forall_f 和 \exists_f 的定义, 等价於以下的定义: The following 2 formulas are from [Abramsky and Tzevelekos

2011]. For all $S \subseteq X, T \subseteq Y$:



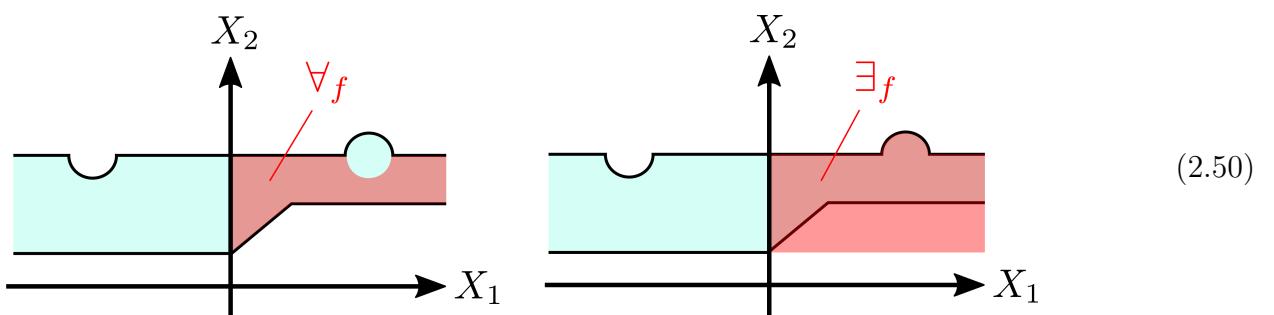
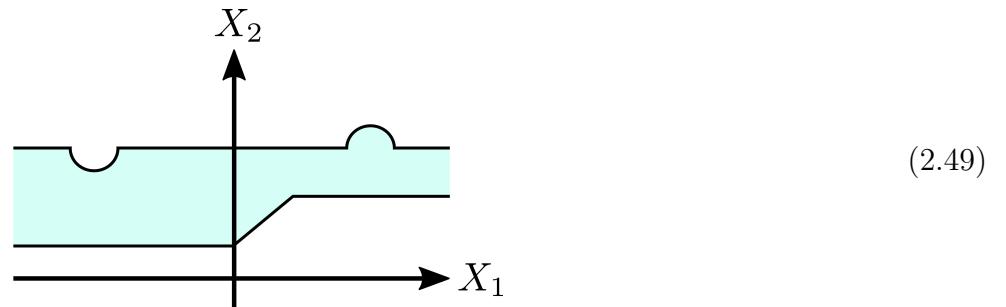
They can be interpreted in a “cylindrical” way: Let $X = X_1 \times X_2$ and $Y = X_2$. Then f is a projection onto the second component, $X_1 \times X_2 \rightarrow X_2$:



Lawvere’s generalization suggests that f need not be a simple projection onto a component dimension (as in cylindric algebra). It seems to work where f is any non-injective function. For example, let

$$f(x_1, x_2) = (|x_1|, x_2) \quad (2.48)$$

where $|\cdot|$ is the absolute value function. Then we have the following behavior:



[Lawvere and Rosebrugh 2003], [Lawvere 1963]

2.12.3 Cut elimination

2.13 量子逻辑

2.14 项重写系统

2.15 图重写系统, 超图

Hypergraph 似乎和 simplicial complex 同构。

Bibliography

- Abramsky and Tzevelekos (2011). Introduction to categories and categorical logic. In: New structures for physics. Ed. by Coecke. Chap. 1.
- Cox, David, John Little, and Donal O'Shea (2007). Ideals, varieties, and algorithms – an Introduction to computational algebraic geometry and commutative algebra. Springer.
- Crole, Roy (1993). Categories for types. Cambridge.
- Ireson-Paine, Jocelyn (2000). Generalisation is an adjunction. URL: <http://www.j-paine.org/generalisation.html>.
- Jacobs, Bart (1999). Categorical logic and type theory. Elsevier.
- Lambek and Scott (1986). Introduction to higher order categorical logic. Cambridge.
- Lawvere (1963). Functorial semantics of algebraic theories. PhD thesis. Columbia university.
- Lawvere and Rosebrugh (2003). Sets for mathematics. Cambridge.
- Sørensen and Urzyczyn (2006). Lectures on the Curry-Howard isomorphism. Elsevier.

3 不确定性

3.1 模糊性	23
3.2 机率	23
3.2.1 Bayesian networks	23
3.3 可信度	23
3.4 推理算法	23
3.4.1 MCMC (Markov chain Monte Carlo)	23

3.1 模糊性

3.2 机率

3.2.1 Bayesian networks

3.3 可信度

3.4 推理算法

3.4.1 MCMC (Markov chain Monte Carlo)

4 神经网络

4.1 神经科学	24
4.1.1 大脑结构	24
4.1.2 神经元	24
4.1.3 神经化学	24
4.2 神经网络的数学	24
4.2.1 非线性分析	24
4.2.2 拓扑度理论	24
4.2.3 同调论 / 奇点理论	24
4.2.4 调和分析	24
4.3 深度学习	24

4.1 神经科学

4.1.1 大脑结构

4.1.2 神经元

4.1.3 神经化学

4.2 神经网络的数学

4.2.1 非线性分析

4.2.2 拓扑度理论

4.2.3 同调论 / 奇点理论

4.2.4 调和分析

4.3 深度学习

5 进化算法

5.1 自然进化历史

5.2 进化算子及其算子谱

6 强化学习

6.1 控制论, 微分几何

6.2 最优化

Part II

功能组別

7 模式识别

7.1 视觉

8 信念修正，真理维修

9 归纳学习

9.1 基於逻辑的归纳学习

10 自然语言

10.1 语法

10.2 语义

10.2.1 Abduction-as-interpretation

10.2.2 Montague 语法

10.2.3 Categorial 语法

11 计划

11.1 自动程式生成

Part III

系统

12 认知系统架构

13 记忆系统

13.1 工作记忆

13.2 事件记忆

14 实践

14.1 道德问题

14.2 商业化

致谢

In addition to the people listed on the title page, I'd like to thank the AGI mailing-list participants for years of discussions.