AGI logic tutorial

YKY

June 1, 2022

Summary

- Outlines the mathematical structure of logic as formulated using category theory, the latest and most satisfactory approach so far. This mathematical knowledge may help the AGI designer if they want to endow neural networks with logic structure, if this is deemed desirable.

Contents

0 Background

We want to **train** an intelligent system. Training is a process of **machine learning** or **optimization**. The goal is to maximize the **sum of long-term rewards**:

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

where $R(t)$ = reward at time t. $\int_0^\infty$ represents the **time horizon** for calculating the cumulative reward. (Here I use the differential version, but the actual computation is discrete. They are basically the same, so we don't go into the details.)

Chess beginners often focus on short-term material gains, overlooking possibilities of sacrificing some pieces to achieve checkmate in few moves. This is **unintelligent**. The formula (0.0.1) requires the system to take into account long-term interests, thus forcing it to learn **intelligent** behavior.

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

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

Or expressed in discrete form:

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

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

F is called the transition function. Or more figuratively:

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

Our goal is to **learn** the function F, implemented as a **deep neural network**. F contains all the **knowledge** in the intelligent system.

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 $\vdash$:

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

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

In the following pages, I will outline a complete logical theory for AGI. A central idea of this theory is the 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**:

$$\begin{aligned} &\boxed{\text{mathematics}} \quad \overset{\text{quantum theory}}{\longrightarrow} \quad \boxed{\text{physical reality}} \\ &\boxed{\text{mathematics}} \quad \overset{\text{Curry-Howard theory}}{\longrightarrow} \quad \boxed{\text{logic}} \end{aligned} \tag{2.0.1}$$

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.

Simply put: logic is a **syntax** on the surface, for example the formula $A \Rightarrow B$:

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

And underneath this syntax, there is an **operation**, which can be regarded as executing a **proof**, which maps the proof of A to the proof of B. [*]

In traditional mathematical notation, a function such as $f(x) = x + 2$, is written as:

$$\begin{array}{c} f : \mathbb{R} \longrightarrow \mathbb{R} \\ \hline x \longmapsto x + 2 \end{array} \tag{2.0.3}$$

This is not new. Similarly, a logical formula:

$$x \text{ is even} \Longrightarrow x + 2 \text{ is even} \tag{2.0.4}$$

is also not new. But to regard the **proposition** "x is even" as a **type** or **set** with a **proof witness**, this perspective has new substance:

$$\begin{array}{c} \boxed{x \text{ is even}} \\ \hline \blacksquare \end{array} \tag{2.0.5}$$

This is called the **Brouwer-Heyting-Kolmogorov (BHK) interpretation**. Because of the subtlety of the idea, it has been re-discovered many times. The names can include: Brouwer-Heyting-Kolmogorov-Schönfinkel -Curry-Meredith-Kleene-Feys-Gödel-Läuchli-Kreisel-Tait-Lawvere-Howard-de Bruijn-Scott-Martin-Lö f-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

[*] To avoid clutter, I will denote all "proof witnesses" as $\blacksquare$, even though each of them is different.

Curry-Howard isomorphism induces an entire enterprise of applying mathematics to the study of logic.

According to HoTT (homotopy type theory), a proposition can be proven **true** or **false**; if it is true, its proofs are all the same, so classical logic propositions can only take the value **true** or **false**. My theory posits that the truth value of fuzzy logic $\in [0, 1]$ because there can be "partial proofs" of fuzzy propositions (see §7).

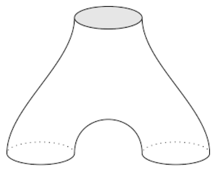
John Baez (1961-)

From another perspective, Curry-Howard isomorphism can be regarded as a duality between **states** (states, such as A) and **transitions** (such as $A \xrightarrow{f} B$). This duality keeps appearing in completely different categories:

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

(2.0.6)

The first two is the Curry-Howard correspondence, the third is added by Lambek, and the rest are from John Baez & M. Stay's paper: *Physics, Topology, Logic and Computation: a Rosetta stone* [2010]. For example, in physics, there is the duality of Hilbert space and operators; In topology, a famous example of **co-bordism** is this "pair of pants":



(2.0.7)

which in String Theory represents the "time course" of the two strings above becoming the string below.

2.1 Type theory

The language that describes **programs** or **computation** is called type theory. For example, the following line is typical in common programming languages:

```
define length(s: String): Integer = { .... }
```
(2.1.1)

It means that length() is a function, that takes an input String, and outputs an Integer.

In mathematics, we denote **functions** by:

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

This expression is actually a general form of type theory:

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

And this notation $t : T$ can actually be written as $t \in T$ (though unorthodox).

In other words, a type is a **set**, and terms are the **elements** in the set.

More generally, a type theory statement can contain type **contexts**:

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

It is like "declaring" the types of variables at the beginning of a program, and then the program can be **assigned** the latter type.

This $\vdash$ process is called **type assignment**, and it is all that type theory does.

λ-calculus

In a program, besides defining its type, you also need to define its **function**. This work is done by λ-calculus.

λ-calculus can define functions without mentioning its "name". For example, in mathematical notation:

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

Its λ-expression is:

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

Note: In the λ-expression, there is no need to mention the "name" of f.

λ-calculus was invented by Alonso Church to study the properties of **substitution** in mathematics. Substitution is something every middle school student knows to do, but is surprisingly troublesome to express in mathematics.

At the same time, Church found that λ-calculus is a "universal" form of computation, which is equivalent to **Turing machines**. John McCarthy, the "Father of AI", used λ-calculus to develop the **Lisp** language, which is the origin of all functional programming languages.

Curry-Howard correspondence

Under Curry-Howard correspondence, the type A is the logic proposition A, and the element in type A (or set A) is its **proof** (or proof witness).

Also, $A \Rightarrow B$ is a logic proposition, which corresponds to the function type $A \to B$, which can also be written as B^A, and the elements in this type or set are certain functions $f : A \to B$. If such functions exist, then type $A \to B$ is "inhabited", in other words $A \Rightarrow B$ has **proof**.

2.2 Intuitionistic logic

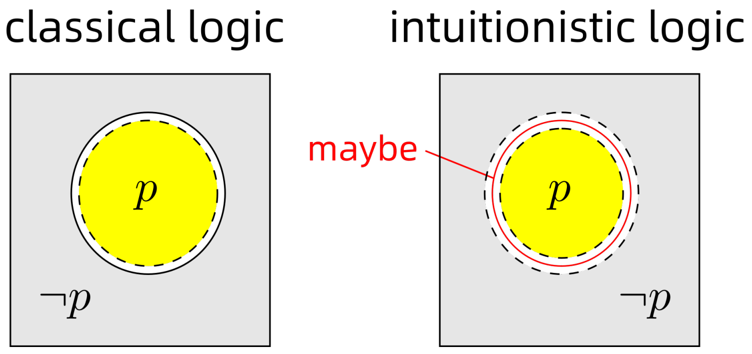
The Curry-Howard isomorphism reveals the relationship between type theory and **intuitionistic logic**. The characteristic of this logic is that there is no **law of excluded middle** (LEM), or equivalently, **double negation**, that is, $\neg\neg p \Rightarrow p$.

The law of excluded middle says $p \vee \neg p$ is a tautology. But in intuitionistic logic, $p \vee \neg p$ means the proof of p **or** the proof of $\neg p$. Sometimes neither of them is known (for example, no proof has been found yet, or it is impossible to find such proofs).

Incidentally, people are surprised to find that under intuitionistic logic, the axiom of choice $\Rightarrow$ law of excluded middle. In other words, the axiom of choice and intuitionism have inherent contradictions.

Topological interpretation

In topology, **open sets** are generally used to represent subsets in space (this originated from the Hausdorff period). But the **complement** $\overline{p}$ of p is not open, so we should define $\neg p$ as the **interior** of the complement of p, that is, $\neg p \triangleq \overline{p}^{\circ}$. Thus $p \cup \neg p \neq$ Universe: [*]



(2.2.1)

2.3 Higher-order logic

Propositional logic deals only with propositions, ignoring any **internal structure** of propositions.

Assuming that p, q are propositions, the basic operations in propositional logic are $p \wedge q, p \vee q, p \Rightarrow q, \neg p$.

First-order logic allows to construct propositions like this:

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

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

Predicates are ”propositions with holes”. They become complete propositions after being filled in with objects. Similarly, there can be predicates of **multiple arity**, for example:

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

The adjective “first-order” is related to ∀ and ∃. These **quantifiers** can **act on** all first-class objects, for example (*everyone loves Mary*):

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

But first-order logic does not allow quantifiers to act on predicates; That requires second-order logic.

An example of second-order logic is “*Napoleon has all the qualities that a good general should have*”:

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

Note that p is quantified over predicates.

2.4 Old-style logic with type theory

The history of Type theory can be traced back even earlier. It began with Russell's attempt to resolve **logic paradoxes**, for example: “*Does a barber who only shaves people who don't shave themselves shave himself?*” The root of these logic paradoxes stems from: definition of something that **refers to** the thing itself. Such a problematic definition is called **impredicative**. In order to avoid bad definitions, everything's type must be ”declared” before it appears. This is the original purpose of type theory.

Before Curry-Howard isomorphism was taken seriously, there was a simpler way to define logic with type theory. In this approach, the logic propositions $p, q, p \wedge q$ etc. are **defined directly** by terms, rather than via “propositions = types, proofs = terms” as in Curry-Howard.

In this case, type theory deals with the aspect of (first- or higher-order) predicates. For example, in:

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

IsHuman is a function term, it inputs an object, and outputs the truth value $\in \Omega = \{\top, \bot\}$ of whether it is a "person". Therefore IsHuman is a term of type $\text{Obj} \to \Omega$.

This approach has no room for Curry-Howard isomorphism. To do the latter, we would need Martin-Löf type theory....

2.5 Martin-Löf type theory

According to Curry-Howard, $A \Rightarrow B$ is a logic proposition and therefore a type:

$$\begin{tikzcd} \overbrace{\text{Human (Socrates)}}^{A} \arrow[d] \arrow[r, phantom, "\Rightarrow"] & \overbrace{\text{Mortal (Socrates)}}^{B} \arrow[d] \\ \Omega & \Omega \end{tikzcd} \tag{2.5.1}$$

On the other hand, Human() and Mortal() are predicates and they also need type theory to form propositions. They are also types. The two levels red $\to$ and $\Rightarrow$ are completely different, but they are forced to live together because of Curry-Howard. It seems that type theory is "multi-tasking".

In "simple" type theory one can construct:

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

They correspond to the intuitionistic logic $\vee, \wedge, \Rightarrow$. These are at the level of **propositional logic** and they have "exhausted the bag of tricks" in type theory.

However, Human(Socrates) is also a proposition composed of Human() and Socrates. This requires to use an arrow $\to$, but there are no more arrows available.

The solution proposed by Martin-Löf is to introduce new **type constructors**:

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

The type of B in the dependent sum $\sum\limits_A B$ depends on A. The sum of the entire family of A (indexed by B) is similar to the product $A \times B$.

The type of B in the dependent product $\prod\limits_A B$ depends on A. The product of the entire family of A is similar to the exponentiation B^A.

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

A very beautiful result is: If we use $\sum_A B$ and $\prod_A B$ to define logic propositions, these types, if inhabited, correspond to $\exists A.B(A)$ and $\forall A.B(A)$, respectively. This is because: If $A \times B$ is inhabited, it means there **exists** at least one $B(A)$; and if B^A is inhabited, there is a function that sends **arbitrary** A's to B's.

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

Many people know that in classical logic $\wedge, \vee$ correspond to the **arithmetic operations** $\times, +$ (they also correspond to $\min, \max$ in fuzzy logic.) Historically, that was the reason why George Boole tried to formulate **logic** as some kind of **algebra**.

[*] Note that "objects" here mean logic objects, not objects in category theory.

What fewer people know is that $A \Rightarrow B$ also corresponds to B^A: [*]

A	B	$A \Rightarrow B$	B^A
0	0	1	$0^0 = 1$
0	1	1	$1^0 = 1$
1	0	0	$0^1 = 0$
1	1	1	$1^1 = 1$

(2.6.1)

where 0^0 is "indeterminate", but according to combinatorics convention, it can be defined as 1.

This surprising "coincidence" seems to prove once again that Curry-Howard correspondence is correct; In particular, it means that $\Rightarrow$ should be interpreted as (function), the so-called "functional interpretation of logical deduction."

2.7 The problem of material implication

To observe in more detail, the truth values of A and B in table (2.6.1) can be regarded as whether their types have **inhabitants**. A type's inhabitant is its proof $\blacksquare$, whereas $\varnothing$ means no proof. More abstractly: the truth value of the proposition A = the **cardinality** of the type A as a set; $|A|$. In this way, the true value of $A \Rightarrow B$ is $|B^A|$, that is, the **number** of maps from $\{\blacksquare\}$ or $\varnothing$ to $\{\blacksquare\}$ or $\varnothing$, and such maps is the empty set only in the case $\{\blacksquare\} \mapsto \varnothing$ (which is impossible).

This observation may be extended to fuzzy logic (§7.1) and strict implication $A \strictif B$ (§6.5).

However, material implication seems to lead to a certain fallacy:

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

For example,

$$\text{seeing black cat} \wedge \text{car accident} \quad \vdash \quad \text{seeing black cat} \Rightarrow \text{car accident} \tag{2.7.2}$$

That is to say, any two **coincidental** occurrences will lead to a conclusion similar to **causality**, even though this causal relationship may not exist. This fallacy

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

seems stem from the confusion of **cases at different times**. In other words: We should verify all the cases in table (2.6.1) before drawing the conclusion $A \Rightarrow B$; However, according to classical logic, only one case is needed to establish that $A \Rightarrow B$.

So, why is this flaw not discovered in classic logic AI systems?

The principle of the so-called **inductive learning of logic rules** is based on the following "generative model":

$$\text{generators} \;\overset{\text{generate}}{\vdash\!\!\relbar\!\!\relbar\!\!\relbar\!\!\relbar}\; \text{data of the world} \tag{2.7.3}$$

The idea behind such "generators" is the same as generators of ideals, groups, function fields, etc. in mathematics.

The purpose of machine learning is to obtain a set of such generators, and the **generating mechanism** is logic derivation, $\vdash$.

However, since the cases (occurring at **different times**) need to be verified during the learning process, the problem of accepting $A \Rightarrow B$ too easily is avoided.

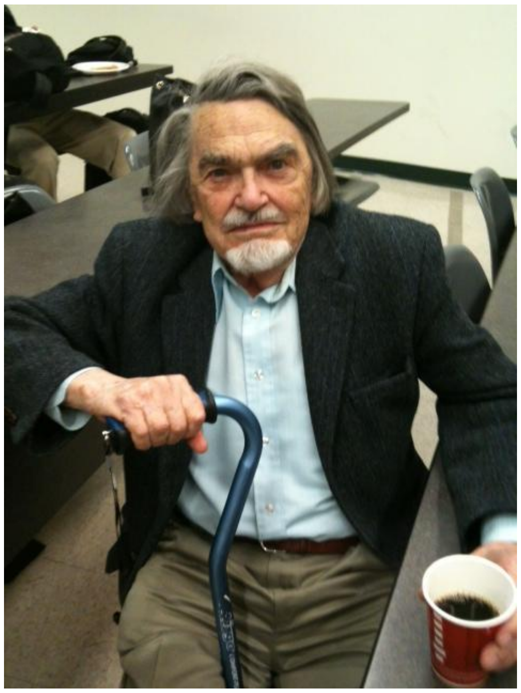
3 Topos theory

3.1 Basics

Making a connection between type theory and category theory, this work was done by Lambek, thus creating a Curry-Howard-Lambek "Trinity":

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

Joachim Lambek (1922-2014)



The significance of topos is that it is a category in which one can perform **logic / set-theoretic operations**. The key is that it can express the concept of **sub-sets**, or more generally called **sub-objects**.

Every topos (C) has a sub-object classifer Ω, so that $X \to \Omega \cong$ sub-objects of$X$. In other words, a subset of X can be represented by the map $X \to \Omega$.

In the topos of **Set**, Ω is a set of **two** elements, which can be written as $\{\top, \bot\}$. Maps like $X \to \Omega$ are **propositions**. For example, if X is a collection of people, then $X \xrightarrow{\text{mathematician}} \Omega$ defines who is a mathematician.

The most important commutative diagram in Topos theory is this:

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

where:

- $X \xrightarrow{!} 1$ is a unique arrow, which maps the set X **entirely** to 1. 1 is the **terminal object**, which is defined by saying there can only be one arrow leading to it.
- $1 \xrightarrow{\text{true}} \Omega$ picks out $\top$ between $\top$ and $\bot$, so we call it the "true" arrow.

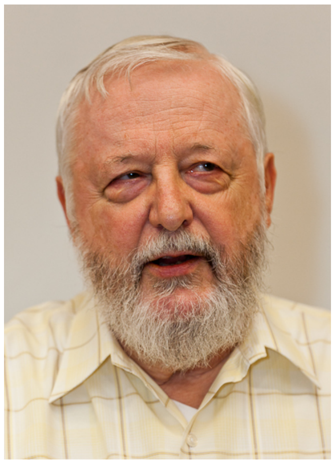
- $X \xrightarrow{m} Y$ is a **monic** arrow; Particularly, in **Set** it is the **inclusion map**, ie $X \xhookrightarrow{m} Y$. It means that X is a **subset** of Y, $X \subseteq Y$.
- $Y \xrightarrow{\chi_m} \Omega$ is the familiar **characteristic function** in set theory. When the element $e \in X \subseteq Y$, $\chi(e)$ takes the value 1, otherwise it is 0. It is the existence of χ_m that makes this diagram commute. χ_m is also denoted $\ulcorner m \urcorner$.

For readers who are not familiar with basic category theory, I highly recommend the book *Conceptual Mathematics*, written in a style even high school students can understand. One of the authors is Lawvere, creator of topos theory.

From the perspective of **Set**, this diagram is easy to understand, but the advantage of topos is that it can **generalize** these logical concepts to a more general category than **Set**.

The importance of the Topos theory is that it uses the language of category to **re-formulate** the entire basis of set theory. In particular, the symbols in logic, such as $P(x), \forall x, \exists x$, seemed impossible to express in category theory. This is Lawvere's amazing achievement.

William Lawvere (1937-)



Look again at the commutative diagram (3.1.2):

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

The $\begin{smallmatrix}1\\ \downarrow\\ \Omega\end{smallmatrix}$ on the right is called the **generic subobject**. Its **pull back** square is indicated by $\lrcorner$. And the $\begin{smallmatrix}X\\ \downarrow\\ Y\end{smallmatrix}$ on the left is a general 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 $p(x_1, x_2, \ldots)$, then

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

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**:

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

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 $\mathrm{Forms}(\bar{X})$ 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:

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

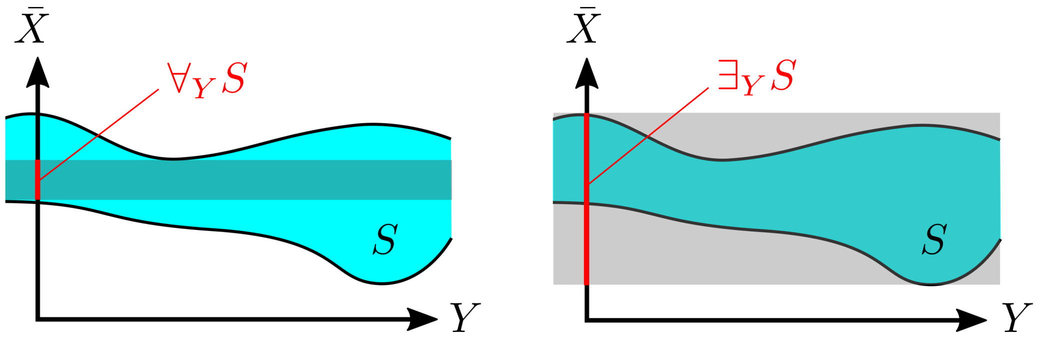
taking each formula $\Phi(\bar{X})$ to itself.

It turns out that ∃ and ∀ are **adjoints** to the map δ:

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

or simply denoted as $\exists \dashv \delta \dashv \forall$. And this makes a lot of sense, because a formula $\Phi(\bar{X}, Y)$, after being quantified by $\forall Y.\ \Phi(\bar{X}, Y)$, becomes a formula that is **independent** of Y.

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:



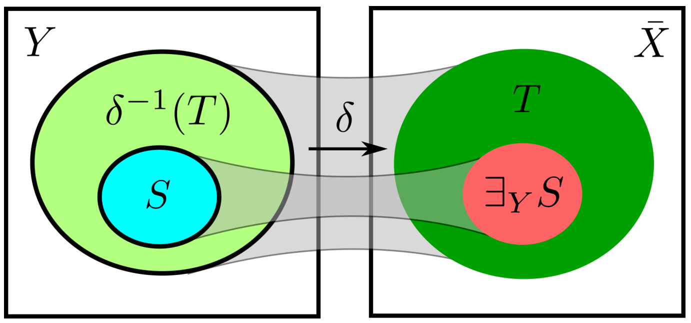
(3.3.3)

Another way is to define with the help of a variable set T [*] (Note that $\bar{X}$ and Y

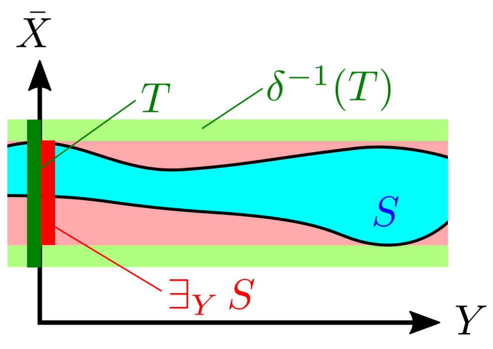
[*] This formula is from [1]

are show as 2D domains below) :

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



Readers familiar with Galois theory may understand adjunctions as a generalization of **Galois connections**. In everyday language: We know S, and would like to define $\exists S$. S lives in the domain Y, $\exists S$ in the domain $\bar{X}$. We seek the help of a "shadow" T in domain $\bar{X}$ to get ahold of $\exists S$, and the shadow's "original" in domain Y would be holding S. The following is a special case where δ is a projection, one that "kills" the dimension Y:

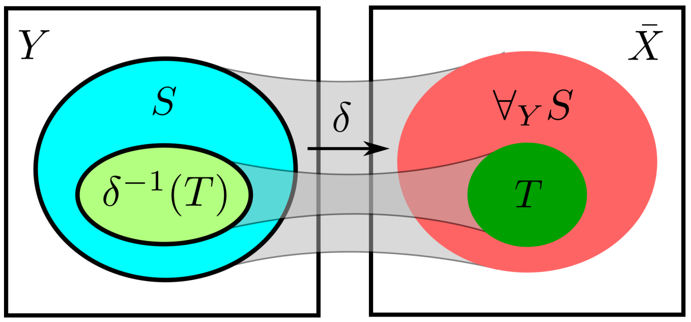


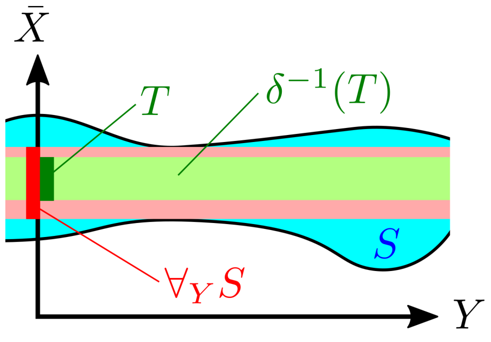
(3.3.5)

Note that in (3.3.5), δ is a projection from $\bar{X} \times Y \to X$, but the δ in (3.3.4) can be **any** map $Y \to X$, and this seems to be the most general definition of $\forall$ and $\exists$. The advantage of using categorical definitions is that they can be easily transferred to other categories, such as Hilbert space.

Similarly we have the definition of $\forall$:

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





(3.3.7)

3.4 ∧ and ⇒ as product-hom adjunction

A logical system with only atomic propositions is "stationary" and cannot draw new conclusions:

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

But if a conditional formula $A \wedge B \wedge C \Rightarrow D$ is added to the left, a new conclusion D can be drawn:

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

In other words, the ⇒ operator is the "fuel" of a logic engine, without which the process of logic inference cannot "run".

The role of $\vdash$ is similar to ⇒, but $\vdash$ is a **meta-logic** symbol, whereas ⇒ is an operator within logic.

In (3.4.2), $A \wedge B \wedge C \Rightarrow D$ led to the appearance of $A \wedge B \wedge C \vdash D$.

Generally speaking, $\Delta \Rightarrow \Gamma$ is a **section** of the mapping $\vdash$ by Δ (a restriction of the $\vdash$ map to the domain Δ). This is very important: a map acts on certain elements, but the elements and the maps are "on the same footing". This is actually a defining characteristic of logic structure.

There is a very important **product-hom adjunction** in topos, which refers to the adjacency between $A \wedge B$ and $A \Rightarrow B$:

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

This is nothing unusual in logic. It means that $A \Rightarrow B$ can replace $A \vdash B$. Since $\vdash$ is a **neural network** in our AI system, this means the "black box" knowledge in the neural network can be "externalized" as **logic propositions**, which is of key importance in intelligent systems, because it means that knowledge can be obtained through *learning via language* (Although one does not necessarily need category theory to see it.)

3.5 Classifying topos $\leftrightharpoons$ internal language

To understand a **category**, the most important questions are: What are its objects? What are its morphisms?

The correspondence given by Lambek is:

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

We have the following transformations between two formalisms:

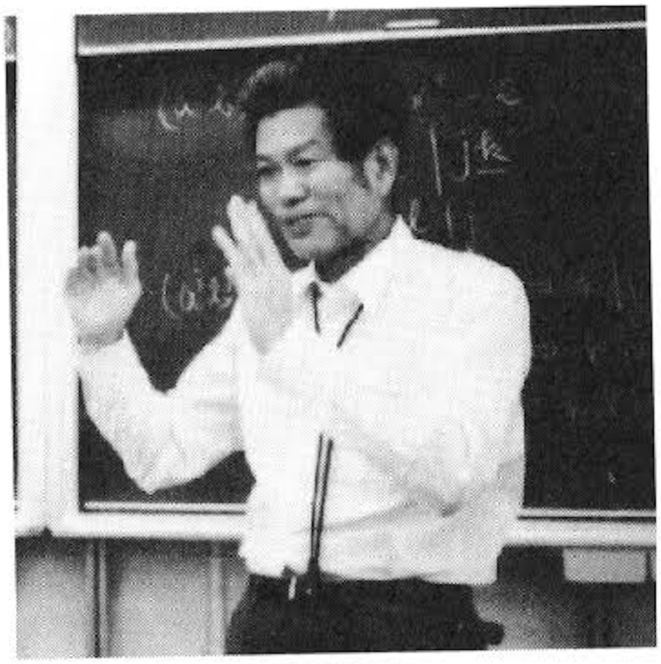
$$\begin{tikzcd} \boxed{\text{topos}}\ \mathcal{C} \arrow[r, shift left, "\text{internal language}"] & T\ \boxed{\text{type theory}} \arrow[l, shift left, "\text{classifying topos}"] \end{tikzcd} . \tag{3.5.1}$$

In other words,

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

3.6 Yoneda lemma

Nobuo Yoneda (1930-1996)



In a category $\mathcal{C}$, consider the morphisms from one object A to other objects, $A \to \bullet$. This is like "seeing" other objects through A.

Example: In **Set**, $1 \to X$ means to "look at" other objects through 1, and what you see is the set of **elements**.

Example: A mapping $\mathbb{R} \to X$ is a **curve** in the space X. It can be said that $\mathbb{R}$ "sees" curves.

Example: In the ordered set $(\mathbb{R}, \leq)$ the object $0 \to x$ can "see" whether x is **positive**.

Similarly, we can consider the dual case, $\bullet \to A$ is how other objects "see" A.

Example: In **Set**, $X \to 2$ is the way other objects "see" 2 and what you get is the **subsets** of X, $\wp(X)$.

Example: In **Top**, 2 contains an open set and a closed set, $X \to 2$ sees the **open subsets** of X, 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 denotes $\mathcal{A}(A, -)$; This is a set.

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 $\mathcal{A}$:

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

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

$$\begin{tikzcd}
\mathcal{A}^{\text{op}} \arrow[r, bend left, "H_A"{name=U}] \arrow[r, bend right, "X"'{name=D}] & \mathbf{Set}
\arrow[Rightarrow, from=U, to=D, shorten=2pt]
\end{tikzcd} \tag{3.6.3}$$

Yoneda lemma:

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

{ 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

(I thought this could be an application of Yoneda lemma, but according to some experts on MathOverflow, the resemblance is only superficial.)

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:

$$\begin{tikzcd}
a \arrow[rrrr, bend left] & {\color{red}\cdot} \arrow[rrrr, bend right, red] & b \arrow[rrrr, bend left] & \longmapsto & \llbracket a \rrbracket & {\color{red}\cdot} & \llbracket b \rrbracket
\end{tikzcd} \tag{3.7.1}$$

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 $\boldsymbol{a \cdot b}$ 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

A logic proposition ϕ can be regarded as a function from a certain domain $A \xrightarrow{\phi} \Omega$, where $\Omega = \{\top, \bot\}$.

Another way to put it is that the proposition $\phi(x)$ is true, where x is an **element** of A. In category theory, we use the terminal object 1 to "pick out" elements of A, as follows:

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

In **Set**, any map from 1, $x : 1 \to A$, can be directly regarded as an "element" of A.

But if we replace 1 with another domain C, as in:

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

Such an $x : C \to A$ is called a **generalized element** of A.

Another terminology is to say that C **forces** $\phi(x)$, notation: $C \Vdash \phi(x)$. This comes from the forcing technique, first invented by Paul Cohen, to solve the Continuum Hypothesis, see §3.11.

Yet another way to put it is that $\phi(x)$ is true **at stage** C. (This comes from possible-world semantics.)

3.9 Internal vs external semantics

Suppose $\phi(\bullet)$ is a predicate, and the domain of ϕ is A. For example, $\phi(x)$ means x is male and $x \in$ people. Then the **extension** of the proposition $\phi(x)$ are the elements that are "male" in the set "people".

Knowing the extension of ϕ allows to determine for each $x \in A$ whether $\phi(x)$ is true or false. In other words, we have a method of **interpretation**.

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 $A_1 \times ... \times A_n$.

This approach is called **internal semantics**. (Note that the idea of "extension" is "internal", which is a bit confusing.)

The other approach is to specify which generalized elements satisfy a certain predicate. The latter is called **external semantics**:

3.10 Kripke-Joyal / external semantics

External semantics describe which generalized elements satisfy each formula.

"Generalized element" means $I \xrightarrow{a} A \xrightarrow{\phi} \Omega$, denoted as $a \Vdash \phi$.

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

3.11 Cohen's method of forcing

In mathematical logic / set theory, forcing refers to some **conditions** added to a certain set G, saying that certain elements belong or not belong to G. Such conditions are expressed in logic, for example:

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

is a condition.

For example, a condition c can "force" the set G to have at least 100 prime numbers, or at least 10,000 prime numbers. The notation is $c \Vdash P$, where P is a logic proposition.

Unlike set theory, topos theory allows more **models**, which is the reason behind Cohen's proof's success.

The following content is unrelated to AGI, but is included for mathematical interest.

Continuum hypothesis (CH):

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

This says that the cardinality of the continuum $[0, 1]$, $2^{\aleph_0}$, comes immediately after the cardinality of the countable set.

In 1878, Cantor proposed CH
In 1900, Hilbert listed the continuum hypothesis as the first of his "23 Problems"
    Hilbert gave a proof, but it contained an error

In 1938, Gödel proved that ZF + CH is consistent,

in other words: ZF cannot disprove CH

In 1963, Paul Cohen proved that ZF cannot prove CH

The method he used is called "forcing".

Paul Cohen (1934-2007)

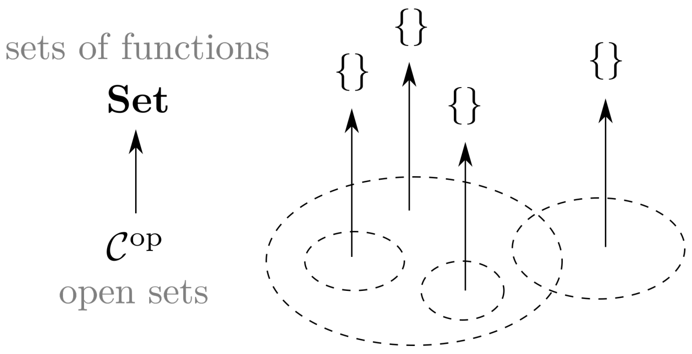
3.12 Sheaves

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 $\mathcal{C}$.

A pre-sheaf must meet the sheaf condition to become a sheaf. This is a typical **gluing condition**, which says that in the overlapping neighborhoods of $U_i \cap U_j$, the sections of the presheaf are **consistent**.

For $\mathcal{C}^{\text{op}}$ = the topology of open sets, I think the following figure is helpful to understanding:



(3.12.1)

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

$$\begin{tikzcd}
\boxed{\mathbf{Set}\text{ topos}} \arrow[d, dash] \\
\{*\}
\end{tikzcd} \tag{3.12.2}$$

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 $\begin{array}{c}\mathbf{Pred}\\ \downarrow\\ \mathbf{Set}\end{array}$.

For the time being, I don't know the significance of sheaf theory to AGI.

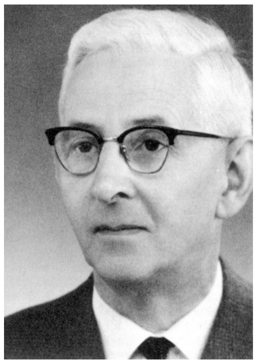
3.13 Kleene realizability

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)

In 1930, Heyting gave a kind of axiomatization of constructive mathematics, called **intuitionistic logic** (IL). Heyting algebra is an **algebraic model** of IL, just as Boolean algebra is an algebraic model of classical logic. (Heyting algebra is to intuitionistic logic what Boolean algebra is to classical logic.)

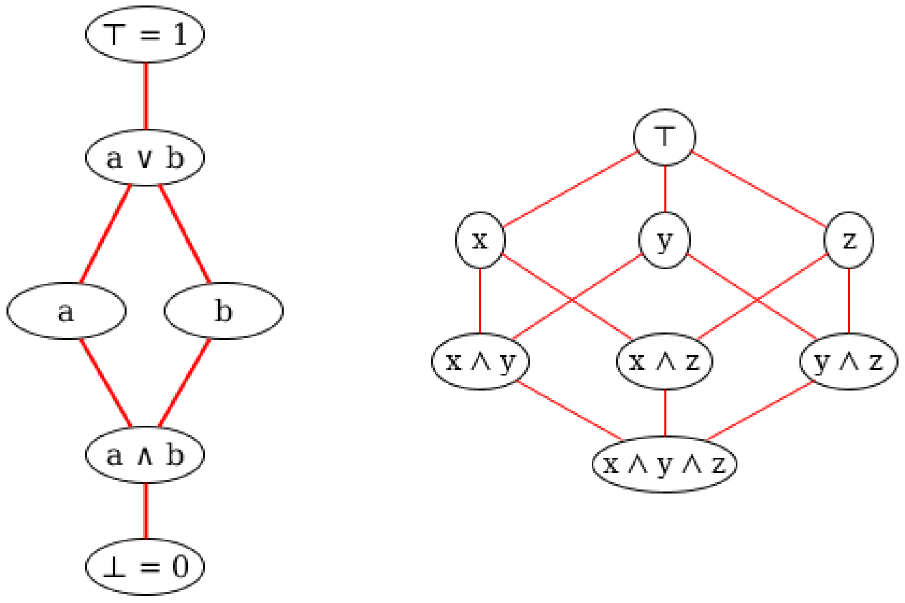
Reference: [**Dunn2001**]

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

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

the **implication order** ($\Rightarrow$) among such propositions. For example, the following are Boolean lattices of 2 and 3 elements [*] :

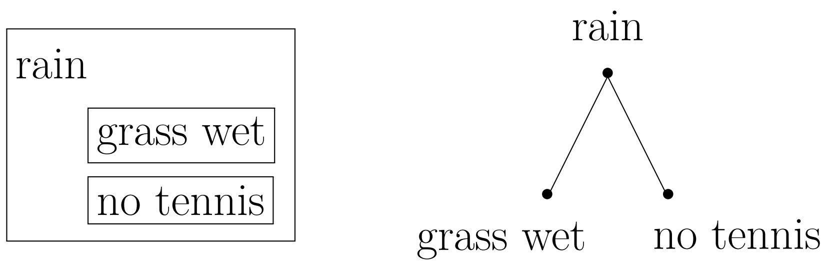


(4.1.2)

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:

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

The following **topological** model and **lattice** model are both models of Heyting algebra:



(4.1.4)

The two are equivalent from **Stone duality** (every Boolean algebra is isomorphic to a topology of open sets), and then generalized to **Priestley** topological duality, etc., all of which are similar .

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 $\vee$.

Note: rainy $\rightarrowtriangle$ grass wet is a Heyting implication. This $\rightarrowtriangle$ arrow does not mean the **true value** of "rainy" is larger than that of "grass wet".

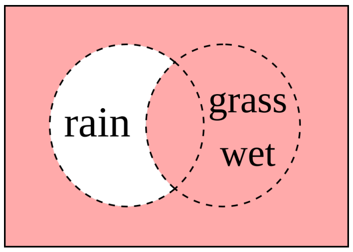
The Heyting implication $a \rightarrowtriangle b$ exists for all elements a, b, x such that:

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

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

The following example may illustrate the definition of Heyting implication $\rightarrowtriangle$:

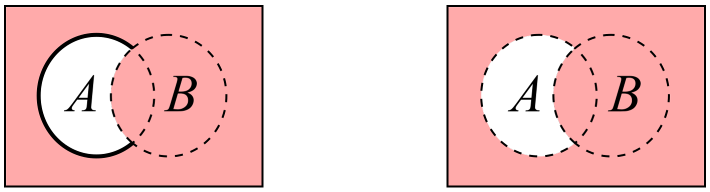
$$\forall X. \quad X \subseteq (\text{rain} \rightarrowtriangle \text{grass wet}) \quad \text{iff} \quad (X \wedge \text{rain}) \subseteq \text{grass wet} \tag{4.1.7}$$



(4.1.8)

The classical $\Rightarrow$ and the intuitive $\rightarrowtriangle$ have only minor differences on the **boundary**:

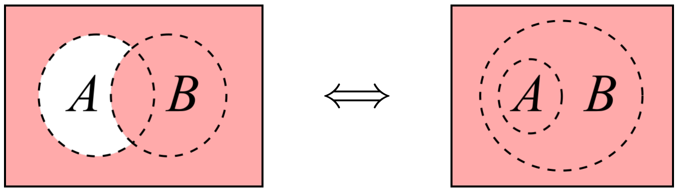
classical $A \Rightarrow B$ Heyting $A \rightarrowtriangle B$



(4.1.9)

Regardless of the logic, the following two sides are **equivalent**:

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



(4.1.11)

This can be seen as the result of continuously shrinking and "wiping out" the white part.

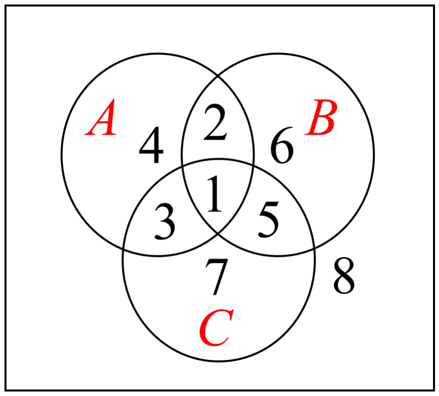
According to Stone duality, since Boolean or Heyting algebra can be used for logical derivation, toplogy of (open) sets can also be used for logical derivation. The basis is the equivalence:

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

This kind of topology can be regarded as a form of **truth table**, as shown in the following figure, where each **area** represents a **row** of the truth table:

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

$\Longleftrightarrow$



(4.1.13)

All this seems to show that Heyting algebra and the more familiar Boolean algebra actually have a lot in common.

Another insight is that the material implication problem in §2.7 still exists in intuitionistic logic.

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":

$$A \prec\!\!\!\!\!\!\;\; 3\, B \quad \equiv \quad \Box(A \Rightarrow B) \tag{4.1.15}$$

The two can be regarded as equivalent via:

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

The question is what are the benefits of defining Heyting implication by possible worlds semantics? It may be more reasonable from the perspective of machine learning. But it also seems to contain classical implication, so is this cyclic?

5 Logic's connections with algebras

[These sections will be expanded later...]

5.1 Gröbner bases

In the book *"Term Rewriting and All That"* [1998, Ch. 8], it is mentioned that the Knuth-Bendix **completion procedure** [1970, first correctness proof by Huet in 1981] for rewriting logic is roughly the same as **Buchberger's algorithm** [1965] for finding Gröbner bases for polynomial rings. The 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 [first discovered by Hironaka in 1964] 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.

5.2 BCI / BCK algebras

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, K, etc. These algebras are non-associative but they also have close connections with associative algebras....

Reference: [2]

5.3 Representation theory

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, and 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. In other words, a representation of a quiver Q is an association of an R-module to each vertex of Q, and a morphism between each module for each arrow.

But this technique only works for **associative** algebras.

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 **Sheaf**(X) is a **sheaf morphism** $\Box : \Omega \to \Omega$ satisfying 3 conditions, $\forall U \subseteq X$ and $p, q \in \Omega(U)$:

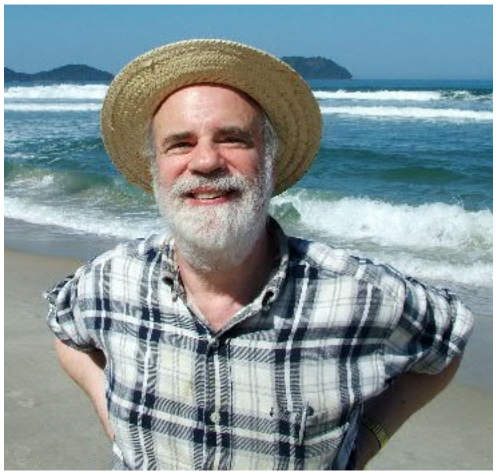
$$\begin{array}{ll} \text{a)} & p \leq \Box(p) \\ \text{b)} & (\Box;\Box)(p) \leq \Box(p) \\ \text{c)} & \Box(p \wedge q) = \Box(p) \wedge \Box(q) \end{array} \tag{6.0.1}$$

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

To interpret modal logic, one introduces a frame $F = \langle W, R, D, H \rangle$, where:

- 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 $\Box$:

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

This involves quantification over all w's.

The point is, what data are needed to do inference?

Obviously it is necessary to decide whether p is true in a certain w, and even to decide if $\forall w.p[w]$.

The latter seems to need to put all relevant possible worlds (or at least their summary) into working memory and then 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 describes implication as it actually "happens", which dictates that $A \Rightarrow B$ is equivalent to $\neg A \vee B$, and the truth table is as follows:

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

(6.5.1)

The concept of material implication has always been controversial. In particular, when the premise is false, the implication is always true:

$$\text{Switzerland is in Africa} \Rightarrow \text{pigs can fly} \tag{6.5.2}$$

By observing the truth table, we can find that each column of it actually represents a **possible world**, and these possible worlds will not happen at the same time. In other words, material implication and strict implication are originally the same, but the former hides the semantics of the possible world "behind the scenes."

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.

The following sections contain tentative ideas, including my own "thinking aloud"

English translation proofread up to here.

7 Fuzzy logic

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

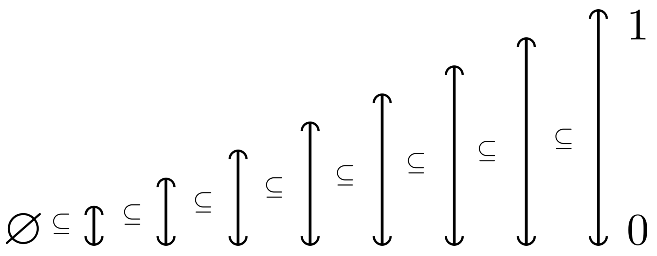
To develop a "fuzzy topos" theory, one requirement is to make the following diagram commute:

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

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:

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

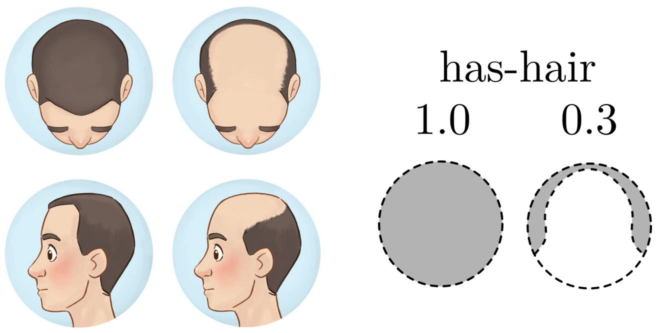
The unit interval $[0,1]$ as Ω can be understood as a lattice, ie, ordered open sets:



(7.0.3)

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

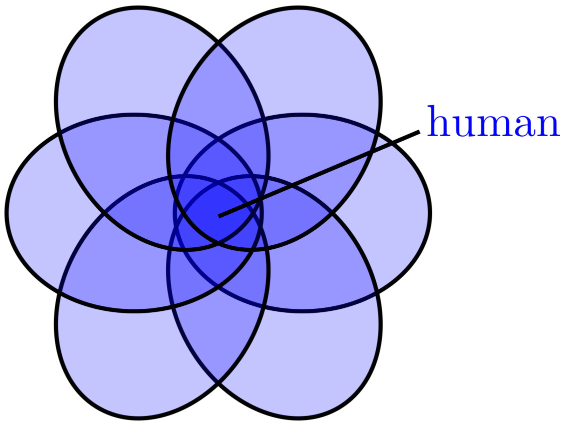
Even more figuratively, we can actually think of the human head with a topology of open sets:



(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.

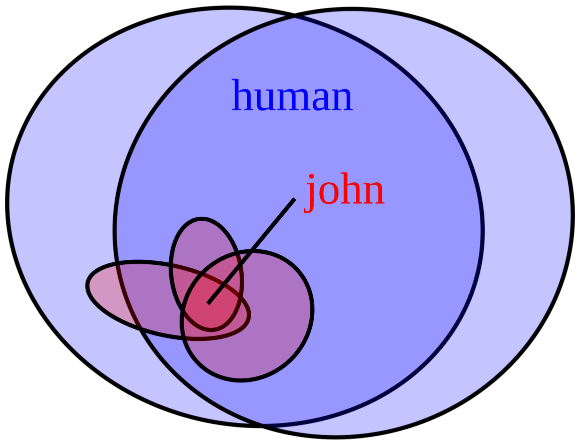
For example, the concept of "person" is represented by several **properties** (properties):



(7.0.5)

For example, humans are a kind of animals, humans have wisdom, humans have certain emotions, humans have certain moral concepts, etc. These attributes are the **intension** of the concept of "person". But the **extension** of the concept of "person" is the **element** of this **set**, such as John, Mary, etc.

And the attribute of "John" is more **special** (specific):



(7.0.6)

The problem is how to decide "John is human" = human(john) The fuzzy truth of this proposition value.

Or use an example that is easier for a man to understand: When we say "Marilyn Monroe is sexy", the fuzzy truth value of this proposition is a certain **commonality between "Marilyn's attributes" and "sexy attributes"** Measurement. This definition of truth value is based on **(**intension) rather than extension.

But dually, it is not impossible to define the truth value from the perspective of **extension**:

- Each fuzzy proposition is regarded as a set, which contains the "proof" of the proposition
- For example, the set of "human beings" is a proof of the proposition that "there are human beings"
- And, "mathematician" as a subset of "human beings" is equivalent to the proof of the proposition "Everyone is a mathematician"

But in some cases, "A is B" but A is not a **set** (for example, John or Marilyn is not a set), the interpretation of extensional does not apply.

Regardless of intensional or extensional, this interpretation method is in harmony with HoTT theory (§8): the **proof** of the concept "John" is $J_1 \wedge J_2 ... \wedge J_n \Rightarrow$ John, where J_i is John's **attribute**. Similarly, the proof of "handsome" is $H_1 \wedge H_2 ... \wedge H_n \Rightarrow$ handsome. Therefore, the fuzzy truth value of "John is handsome" is a measure of the commonality (such as intersection or union) of $\{J_i\}$ and $\{H_i\}$.

There is another problem with the fuzzy truth value: The **attribute** of an object can be increased or decreased at will to a certain extent, so purely calculating the **number** of the attribute cannot objectively determine the truth value. It seems that each attribute There should be some kind of **measure** (measure) or **weight** (weight), so that when the description of the attribute is changed, the fuzzy truth value remains **unchanged**.

What is a proof witness? For example, "the spoon is in the cup" is the result of other visual propositions. Its truth is important, but its "intension" is even more important. In other words, proof objects are the essence in the proof process.

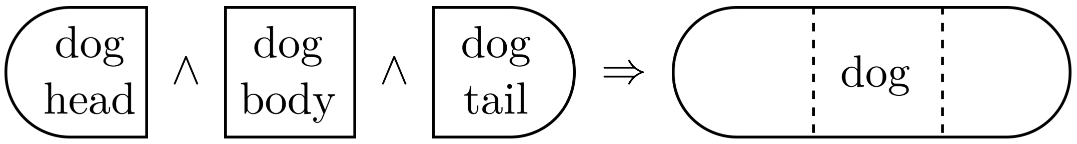
What can this proof object be besides being a syntactic thing? In other words, the **domain** of the map is a proposition, but what is mapped by the map is evidence?

7.1 Fuzzy implication

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

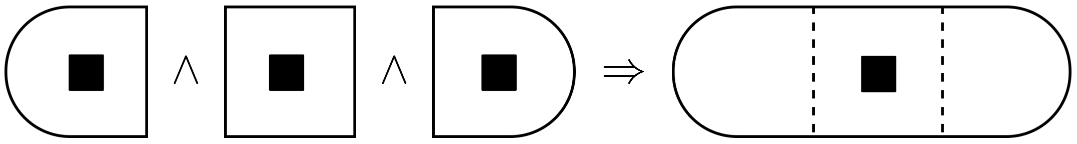
$$\text{dog-head} \land \text{dog-body} \land \text{dog-tail} \Rightarrow \text{dog} \tag{7.1.1}$$

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



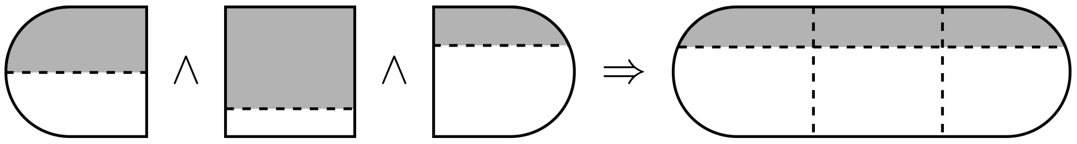
(7.1.2)

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.

This section examines how the true value of fuzzy implication $A \overset{\sim}{\Rightarrow} B$ is determined? The properties of classic material implication have been analyzed in §2.7.

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 $A \overset{\sim}{\Rightarrow} B$ is a proposition, then what are its **proofs**? This should be interpreted as "the number of maps from A to B" The key is to use this to define the truth value of fuzzy implication.

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?

number of maps   fuzzy intension,   number of proofs proof 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.

If A's proof fuzzy   (but B's proof is true), then $A \Rightarrow B$ fuzzy TV   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)

The central idea of HoTT is to regard types as certain spaces, and these spaces can be given topological, especially homotopy structures. In terms of homotopy, the key is to treat the **equal** id_A of the two elements in type A as the **path** of homotopy.

8.1 Why HoTT may be useful

On topos, $A \subseteq B$ or $a \in A$ is a subset relationship, or it can be said to be a topological relationship. On the computer, giving this topology more "space" features, such as vector space, metric space, etc., seems to be very powerful.

But the problem is: if A is a region (with known shape) in topological space, and the position of entity a is also known, then the true or false of the proposition $a \in A$ can also be known immediately. But this is very problematic. For example, the proposition "$e \in ?$ irrational number" requires complex proofs, not instant judgments. Or "OJ Simpson $\in$? Murderer".

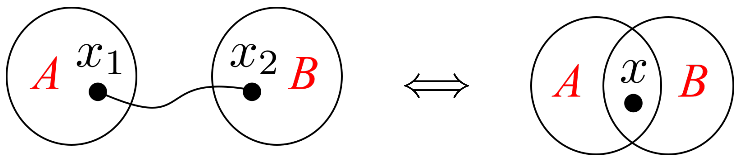
In other words, it seems impossible to realize the propositional space into a space with **specific location**. Therefore, the proposition $a \in A$ can only be expressed as an ordered pair (a, A), and then use **algebraic method** to simulate abstract topology. This is actually what I am in (**??**) The "crude" syntactic method mentioned.

One way to save the "location space" is to allow OJ Simpson ∈ "blacks, men, actors, players", etc., but ensure that he does not belong to other (unverified) collections, the latter is **unknown** of. But this leads to the verification of potentially infinite sets, which is not feasible.

Unless in the logical space, each set **does not intersect** is guaranteed in advance, and if $x \in A \ \wedge \ x \in B$, set $x_1 \in A,\ x_2\ inB,\ x_1 = x_2$, and x_1, x_2 are **path-connected**. So this requires HoTT.

It is not yet known whether this approach is more efficient than the "crude" approach.

According to paths, there can be the following topological "deformations":



(8.1.1)

But on the right, there will still be a situation where $x \in A$ accidentally causes $x \in B$.

If the object of neural network map F is not ordered pairs $(x \in A)$, what can it be? A **proposition** can be regarded as a set of points x_i connected by paths.

8.2 HoTT levels

Voevodsky proposed these "levels":

...	...
2	2-groupoids
1	groupoids
0	sets
-1	(mere) propositions
-2	contractable spaces

(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 $a \overset{?}{=} b$; 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

At the set level of HoTT, "=" is a predicate, which can be regarded as a fuzzy predicate according to my theory.

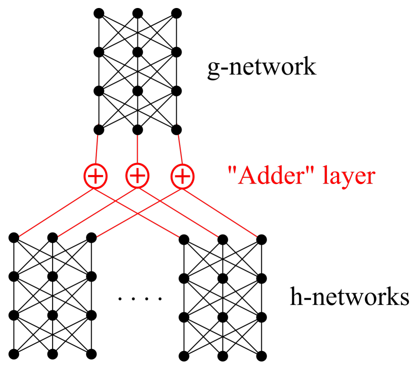
If I did not understand the error, the meaning of univalence axiom seems to be to "force" the fuzzy truth value of = into binary.

9 Transfer to deep learning

This section discusses how to transfer the logic structure mentioned above to the neural network (NN) of deep learning. Generally speaking, it is difficult to impose additional structures on the neural network, because the NN itself already has a very "rigid" structure. A successful example in this direction is CNN (convolutional NN), where convolution $f * g$ is simulated by "weight sharing". But other than that, it is generally difficult to add structure to the NN.

Symmetric NN (permutation invariant)

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



(9.0.2)

Note that under this scheme, NN uses "black box" as building blocks; the internal structure remains unchanged.

The advantage of category theory is that everything is represented by morphisms, compositions, pullbacks, adjunctions, etc.; this approach is easy to implement with NN.

9.1 Propositional aspect

At the level of propositional logic, I chose the simplest characteristic, which is commutativity between propositions:

$$\begin{array}{ccc} A \wedge B & \equiv & B \wedge A \\ \text{it's raining} \wedge \text{lovesick} & \equiv & \text{lovesick} \wedge \text{it's raining} \end{array} \tag{9.1.1}$$

But I did not fully exploit the structure of Heyting algebra or Boolean algebra, because it is foreseeable that it will be extended to fuzzy-probabilistic logic in the future, and the latter structure only requires $\wedge$ and $\Rightarrow$, which is slightly different from binary logic (later Explanation....)

9.2 Predicate aspect

At present, the general deep learning model is relatively simple/crudely applied to the **syntax** level of natural language sentences, for example:

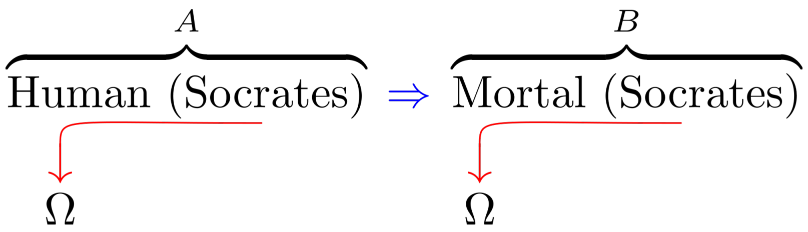
$$\text{“Je} \bullet \text{suis} \bullet \text{étudiant”} \overset{f}{\Longrightarrow} \text{“I} \bullet \text{am} \bullet \text{student”} \tag{9.2.1}$$

The words in the sentence are embed into the vector space in the way of **Word2Vec**. But if Curry-Howard is used, there will be a slight difference, and this subtle difference is mathematically perfect. In fact, will the computer implementation be better? Look again at Curry-Howard correspondence:

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

Here A and B are **logic propositions**, such as "I am a student"; f maps the witness $\blacksquare$ in the A proposition to the B proposition. Note: The information on the **syntax** of "I am a student" is represented by the **domain** and **co-domain** of f.

In this section, we are going to discuss the proposition and its internal implementation on the computer. Look again at the two levels in the picture (2.5.1):



(2.5.1)

red $\to$ is the level formed by **predicates**. A and B are two different **spaces**. In A **inside**, Socrates is also a space, Human is another space, and Human(Socrates) constitutes a new space (through dependent type constructor Σ).

Suppose $X \in \mathcal{U}_1, P \in \mathcal{U}_2, P(X) \in \mathcal{U}_3$, (these $\mathcal{U}_i$ are type universes) , The easiest way on the computer is to:

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

This is consistent with the statement that §2.5 said that the Σ type constructor is essentially a Cartesian product.

At this point, we found that the Curry-Howard approach is exactly the same as the "crude" syntactic approach! A little disappointed, but I hope these beautiful maths will not be completely useless...

9.3 Implementation of topology (points and sets)

Are all propositions embed to the proposition space $\mathbb{P}$rop? Or is there some kind of working-space embedding?

This means that from the long-term memory recall, as long as the inference makes sense is guaranteed, the constants do not have to have a long-term unchanging position.

Or can it be said that the predicates feature of constant determines its position?

But deep learning seems to have an advantage over absolute positions. For example, the location of " ". This means that each different object in memory has its absolute position. Another method is relative position: an object recalled from memory has many predicates attached, but its position is constructed on-demand. What is the difference between these two approaches? It seems that the key is recall mechanism;

But the memory must be a whole with context. The question is whether it can achieve associative block recall.

Is this related to topology? Can LTM be used to reduce the "load" of F? In other words, F is only responsible for the inference of the relative position? If it is the relative position, it seems possible to reshape the topology. But I still can't tell what benefits reshape topology has, and the problems that may cause errors.....

Reshape topology brings generalization but also errors.

The question is, what will happen to F if topology is used? The proposition is $P(a) = 1$. Its information seems to be a tuple anyway (P, a). This will be a map of tuples to tuple, and this is also brutal syntactics. But the point is: $P(a)$ should be a **space**. The structure of this space is through type constructor $\sum_a P(a)$.

This constitutes a space, and F is a mapping from this space to that space. So F is not a mapping of tuples to tuple, but a mapping of spaces to space. But this mapping is a mapping of witness, which is a bit strange. In other words, use memory to remember witnesses, which represent each proposition. Their location is the syntax of the proposition.

There is a witness in the space, its position is exactly the positional tuple of $P(a)$, and it is mapped to the positional tuple of $Q(a)$ at the new position. Isn't this exactly the same as tuple to tuple? ? In other words, no matter how you look at it, the mapping of F **must be from positional tuple to positional tuple**.

$P(a)$ is (P, a), syntactic mapping is inevitable.

So what is topology? The position of a is that a is the position of Atom, which itself can be regarded as a predicate. But, are there any permanent positions for individual constants? If the location is permanent, recall is very easy (direct recall). The question is whether to reshape topology?

Another question is: at the beginning, how is the position of a **new** constant assigned? This is obviously related to **forget**. A simple way is to have **constant space**, and these constants **location is permanent**.

So back to the question: Since there can be tuple-to-tuple, what should topological membership do? It seems to be used to impose topological-metric (regularity) (ie, smoothness).

9.4 Modal aspect

10 Model-based AI

The "logical" AGI model of BERT I proposed is based on logic syntax. In other words, it simulates $\vdash$ with a neural network, and this $\vdash$ is a syntactic consequence. Now consider $\vDash$, which is model- theoretic consequence.

The simplest model is point-set topology. Corresponding as follows:

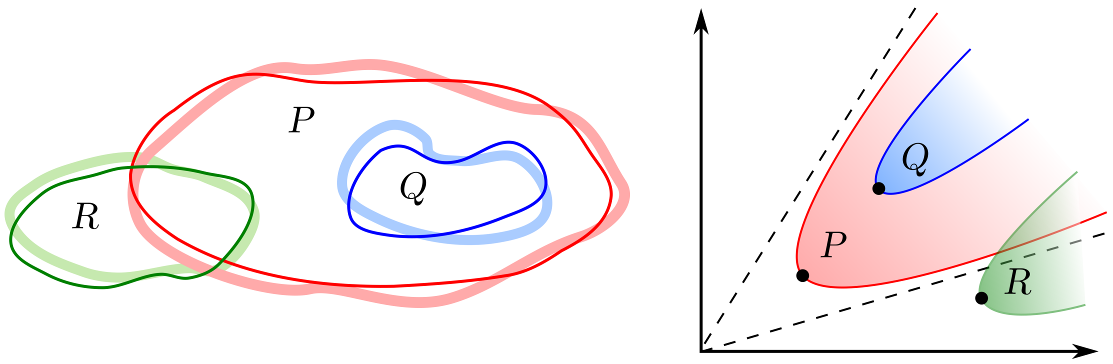
$$\begin{aligned} \text{propositions} &\Leftrightarrow \text{points} \in \text{regions or product of domains} \\ \forall \text{ formulas} &\Leftrightarrow \text{sub-regions} \end{aligned} \tag{10.0.1}$$

One of the main problems of Model-based (logic-based) AI is: In the case of syntax-based, we can introduce a proposition $P(a)$ in working memory, for example, $P =$ "Is a a prime number? ", you can temporarily ignore the truth value of $P(x)$ at other points; but in the case of model-based, the model contains the extension of P ($=$ a point set or area), which means that we immediately It is impractical to know whether each number is prime or not. This problem may be called the "**omniscient predicates**" problem.

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****.*

The second condition is added because the learning algorithm of "classical" logic AI also meets the "depth" condition, but it is not efficient enough.

References

Questions, comments welcome ☺

References

[1] Abramsky and Tzevelekos. "Introduction to categories and categorical logic". In: New structures for physics. Ed. by Coecke. 2011. Chap. 1.

[2] Sørensen and Urzyczyn. Lectures on the Curry-Howard isomorphism. Elsevier, 2006.

[more references will be added...]