Knowledge representation in AI

Yan King Yin general.intelligence@gmail.com

September 13, 2018

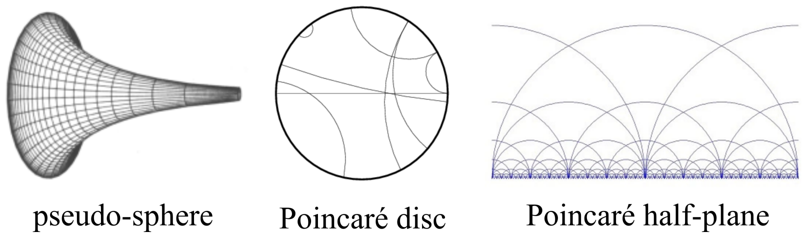
Contents

Abstract

As of now (2018 August), the question of strong AI is no longer whether it is possible or not, but whether one feasible approach is better than another. The tutorial introduces the mathematical theory of knowledge representations, and discusses 3 proposals, respectively based on: A) genetic algorithms; B) neural networks + graphs; C) geometric models.

What is model theory?

For example, **hyperbolic geometry** can be "realized" by the following **models**:



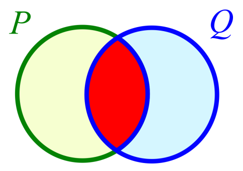
(1)

Models are not unique, there can be many models for a theory.

In mathematical logic, **model theory** studies the **duality** between **syntax** and **models**.

The most classic example is **Stone duality**, also familiar to all of us as "Venn diagrams":

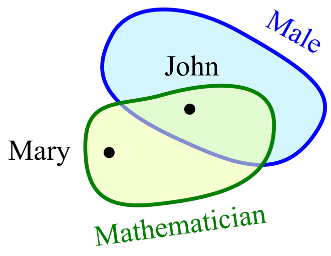
$$P \wedge Q \quad \cong \qquad (2)$$



Stone duality[*] refers to the duality between **Boolean algebras** and **topological spaces**.

Boolean algebra is the same as **propositional logic**, which concerns only with the truth and falsehood of propositions. For example, P = *"It is raining in New York"*, but we cannot "access" the internal constituents of the propositioon, such as *"rain"* and *"New York"*. The critical obstruction to strong AI is the **lifting** from propositional to first-order logic.

First-order logic can be modeled by **sets** and their **elements**. For example John $\in$ Male,   John, Mary $\in$ Mathematician

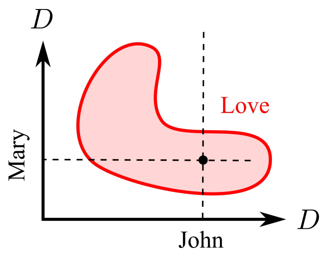


(3)

Whereas **relations** between first-order **objects** in a domain D are repre-

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

sented by **subsets** of the Cartesian product $D \times D$, eg:



(4)

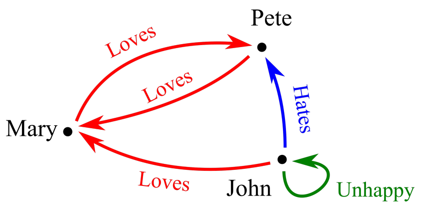
What the fuck's wrong with this line?

For computer science people,

you may be more familiar with **relation graphs**

or **knowledge graphs** such as this one:

What the fuck's wrong with this line?



(5)

This is a **directed multi-graph**, or **quiver**. Quiver is an important structure in algebraic representation theory. The category of quivers $\mathcal{Q}$ is a **topos** (this is introduced in the the basic meaning is it has conditional first-order logic as **model** category [†] .

The above knowledge graph can be simply converted to a collection of **logical children**:

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

(6)

So, logic and graph are basically **equivalent**.

[†]According to [Grilliette 2017], hyper-graph is not topos, multi-graph is not topos, but when they become directed.

If each edge of graph can contain any number of vertices, then there is **hyper-graph**. In other words, each edge of the hypergraph $\in \wp(V)$, V is the vertex set. It can also be said that the hypergraph is the **subset system** of V. For logic, the benefit is: <u>There can be relationships on the relationship</u>. Hypergraph can correspond to the topological **simplicial complex**, and its homology and cohomology can be studied. The Simplicial complex can also correspond to **square-free monomial ideals** one-to-one. Square-free means that the index of x_i can only be 0 or 1. The latter is the scope of the **combina commutative algebra**. For the time being, I don't know if these associations are useful. See [Brown 2013], [Miller and Sturmfels 2005] for details.

A collection of logical expressions is called logical **theory**. A collection of algebraic equations is called **algebraic theory**. For example, you can have the following logical formula ("love is not happy"):

$$\forall x, y.\ \text{Loves}(x, y) \land \neg\text{Loves}(y, x) \to \text{Unhappy}(x) \tag{7}$$

This formula contains universal quantification, so it is not part of the model. Logically, only the collection of **ground sentences** (the expression without variables) can form a model, for example (6) .

fractal

An expression in Logic **theory** can cause many **new** vertices and joins to appear in the model. This is a question of model theory research. In some cases, the model space will have an "infinitely subdivided" fractal structure. For example, each natural number $n \in \mathbb{N}$ has its successor $S(n)$. The existence of this function causes a series of **infinite** vertices in the model space:

$$\text{ffl} \quad \text{ffl} \quad \text{ffl} \quad \text{ffl} \quad \text{ffl} \quad \text{ffl} \quad \text{ffl} \quad \text{ffl} \quad \text{ffl} \ldots.. \tag{8}$$

If you add this **rule**:

$$\forall n \in \mathbb{N}. \quad S(n) \geq n \tag{9}$$

Then immediately generate an infinite number of relationships:

$$\text{ffl} \xleftarrow{\geq} \text{ffl} \xleftarrow{\geq} \text{ffl} \xleftarrow{\geq} \text{ffl} \xleftarrow{\geq} \text{ffl} \xleftarrow{\geq} \text{ffl} \xleftarrow{\geq} \text{ffl} \xleftarrow{\geq} \text{ffl} \ldots.. \tag{10}$$

Although, in **common-sense intelligence**, it seems that this infinite structure is less common, and more is the structure of "shallow". Incidentally, the knowledge representation of classical logic-based AI is split into two parts: **rules** and **facts**. The former is the formula with $\forall$ **variable**, the latter is ground sentences. Rules are stored in KB and facts are stored in **working memory**. The former is a **theory**, which can be thought of as some **"partial" models**. The reason for saying partial is because it does not represent the entire model. In fact, model is a very large thing that cannot be stored in a physical system. Artificial intelligence or the brain can only store certain theories and parts of models. The key issue of artificial intelligence is how to find a good syntax structure to make theory learning faster and more efficient.

This section describes the mathematical structure of a strong artificial intelligence system in a way that is as concise as possible, even if the mathematician without an AI background can understand it.

Revisit **neural network** like this:

$$F(\vec{x}) = \sigma(W_1 \sigma(W_2 ... \sigma(W_L\ \vec{x}))) \tag{11}$$

Its **parameter** collection $\Theta = \{W^{\ell}_{i,j}\} \in \mathbb{R}^m$, where $m = \#$ weights. The purpose of **machine learning** is to find optimal Θ subject to an objective function. In other words, machine learning = **optimization**, which is one of the most basic problems in applied mathematics. When training, give a set of data points (F is a neural network):

$$\boxed{\text{training examples}} \quad e \xrightarrow{F} a \quad \boxed{\text{answers}} \tag{12}$$

The error for each answer is ϵ, and the objective function J is the sum of the errors of many iterations.We want to optimize J by calculating the gradient of J for Θ:

$$\nabla_{\Theta} J := \frac{\partial J}{\partial \Theta} \tag{13}$$

back-propagation gradient descent

$$\tag{14}$$

The feature of the AI learning algorithm is that <u> needs to be optimized on</u> <u>**logical sub**</u>:

$$\begin{array}{rcl} \text{optimization over } \mathbb{R} & & \text{optimization over } \mathcal{L} \\ \Theta \in \mathbb{R}^m & \rightsquigarrow & \Theta \in \mathcal{L} \end{array} \tag{15}$$

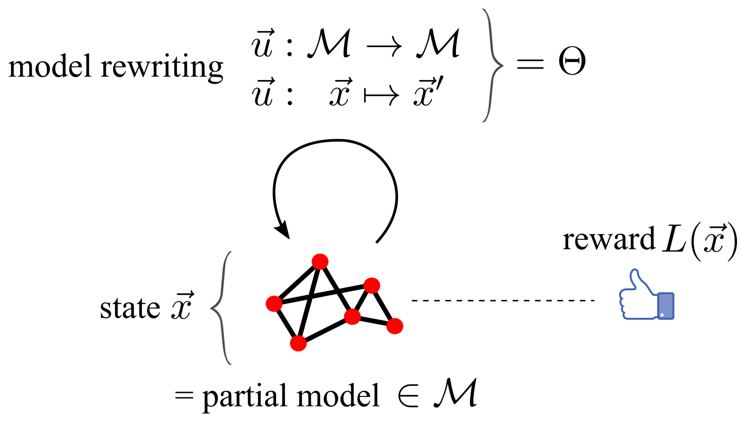
Where $\mathcal{L}$ is some sort of (eg first-order predicate) **logical syntax**, and Θ is a collection of logical children. The solution to this optimization problem Θ^* is an optimal logic theory. The top-level architecture of the system is reinforcement learning, which is dynamic programming. It is a special case of optimization. It can also be called control theory. The **system** it controls is:

$$\vec{x}_{n+1} = \vec{f}(\vec{x}_n, \vec{u}_n) \tag{16}$$

$\vec{x}$ is the system's **status**, and $\vec{u}$ is called control or action. In AI, $\vec{x}$ is the **location** in the "think space", and $\vec{u}$ is the "thinking" step. We want to control $\vec{u}$ so that the system reaches the maximum value of **rewards** in <u>long running</u>:

$$\boxed{\text{total rewards}} \quad J = \sum_n L(\vec{x}) = \int L dt \tag{17}$$

$L \in \mathbb{R}$ is the **instant reward** at the $\vec{x}$ position. Based on historical analysis mechanics, L is also called **Lagrangian**, unit It's energy (but the sign changes, the latter measures the penalty), but note that $\vec{x}$ is a "think space", unlike physical space. I am writing a differential form to make it easier to remember. The system operates as follows:

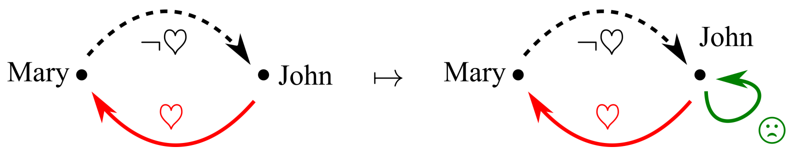

model rewriting
$\vec{u} : \mathcal{M} \to \mathcal{M}$
$\vec{u} : \vec{x} \mapsto \vec{x}'$
$= \Theta$
reward $L(\vec{x})$
state $\vec{x}$
= partial model $\in \mathcal{M}$

(18)

$\vec{u}$ and $\vec{f}$ coincide, the function is to **rewrite** $\vec{x}$:

$$\vec{f}(\vec{x}, \vec{u}) \equiv \vec{u}(\vec{x}) \tag{19}$$

For example, the logic rule “‘ $\Rightarrow$ ” performs the rewriting of the following sub-graph:



(20)

This is the **state transition** $\vec{u} : \vec{x} \mapsto \vec{x}'$, which can also be regarded as the **logical inference** $\vec{u} : \vec{v} \vdash \vec{x}'$, where $\vec{u}$ is the rewriting function or logic rule. Optimization of J acts on top of Θ (ie $\vec{u}$). Note that $\vec{u} \in$ is a function space or logical formula, which is very different from the traditional optimization over $\mathbb{R}$. In classic AI, $\vec{u}(\vec{x}) = \vec{f}(\vec{x})$ is equivalent to deduction $\vec{x} \vdash \vec{x}'$, or forward-chaining (forward some logical conclusions). In classic AI this work is handled by **logic engine**, which contains two algorithms: **unification** (= rule matching) and **resolution** (= proof search). These operations are not visible in our framework because they are included in $\vec{u}$ or $\vec{f}$.

For experts in applied mathematics, the following theory is quite standard. The definition of Hamiltonian in analytical mechanics is:

$$H = L + \frac{\partial J}{\partial \vec{x}} \vec{f} \tag{21}$$

Similarly, the Hamiltonian of the **discrete** system can be defined as:

$$H = L + J(\vec{f}(\vec{x})) \tag{22}$$

Pontryagin[‡]

$$H^* = \inf_u H \quad \text{or} \quad \nabla_{\vec{u}} H^* := \frac{\partial H^*}{\partial \vec{u}} = 0 \tag{23}$$

J T

$$TJ := \inf_u H \tag{24}$$

Then Bellman's optimality condition can be expressed as the following fixed-point form [**Bertsekas2013**]:

$$J^* = TJ^* \tag{25}$$

It is said that the Hamilton-Jacobi method leads to the solution of partial differential equations, which is not particularly effective. In practice, the Pontryagin minimum principle is more useful. There is a gradient $\nabla_{\vec{u}}$ in (23), so it would be useful to find a derivative for $\vec{u}$. If the problem is continuous optimization, you can use non-smooth analysis. The minimum requirement is that there is norm (*ie*, Hilbert space or Banach space) on the domain, you can use the method such as proximal gradient. The key issue is: Θ belongs to a domain other than the traditional $\mathbb{R}^n$.

Plan 0: geometric models

$$\mathcal{L} \quad \text{model} \tag{26}$$

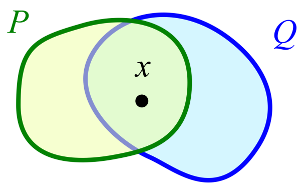
There can be many different variants, for example:

$$\begin{tikzcd} \mathcal{L} \arrow[d] & \approx & \text{graph re-writer} \arrow[d] \\ \text{partial model } \mathcal{M} & & \text{graph model} \end{tikzcd} \tag{27}$$

[‡]Lev Pontryagin (1908-1988) Soviet mathematician. Blind since the age of 14, he made major discoveries in a number of fields including algebraic topology and differential topology.

Consider a model I call **geometric model**, for example:

$$P(x) \wedge Q(x) \quad \cong \tag{28}$$

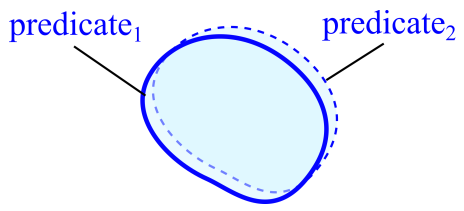


These geometric "**regions**" are easy to do with neural networks. If $\vec{F}$ = neural network, then:

$$\vec{F}(\vec{x}) = 0 \tag{29}$$

hyper-surface of co-dimension 1 > 0 < 0

One effect that you hope to achieve (but perhaps hard to do) is: a logical formula can be deformed into another different expression. This is not possible in classic logic, such as Love(x, y) and Like(x, y), although the meaning is close, but from one to the other must be discrete Jumping. As shown below, two approximate regions:



(30)

distinct objects representation

References

Brown (2013). Discrete structures and their interactions. CRC Press.

Grilliette (2017). "A Functorial Link between Quivers and Hypergraphs". In: URL: `https://www.researchgate.net/publication/305787097_A_Functorial_Link_between_Quivers_and_Hypergraphs`.

Miller and Sturmfels (2005). Combinatorial commutative algebra. GTM 227.