Why is the Mind a Self-programming System ?

Abstract

How the mind works have always been one of the most important and fascinating puzzles. For a long time, countless scholars have devoted themselves to this research, but there has no decisive progress been made yet. We believe that the fundamental reason for this situation is that the research paradigm learned from the study of objective sciences such as physics is not applicable to the study of the mind. In this paper, we suggest that the proper way to study the mind is to think of it as a computational system that can program itself. In this self-programming system, there are two key components: storage structure and bootstrap program. In this paper, we only focus on analyzing the principle of the storage structure. Although this paper does not give a full mechanism of the self-programming system, the mere study of the storage structure can generate some important findings. Specifically, by using the storage structure, the problem of induction can be solved; And the mechanism of deduction and causality can easily be explained. Some of our findings sharply contrast with the existing theories. For example, we argue that the validity of induction is attributed to the intention of self-programming systems, not because it can be justified as mainstream inductivists try to prove.

Keywords: Problem of induction, Causality, Deduction,

1. Background and Methodology

Understanding how the mind works has always been one of the most important and fascinating problems of human beings. Scholars in the past have proposed lots of theories to try to explain the mechanism of the mind, but the way they have been advocated are all based on such a process: 1) Discovering a phenomenon; 2) Proposing a theory to explain this phenomenon; 3) Verifying the theory in more situations. This process is actually

derived from the scientific paradigm of studying the objective world, such as physics and chemistry. However, is this approach suitable for research of the mind? The answer may not be so.

In the past few decades, thanks to the influence of Turing machines and subsequent general-purpose computers, researchers have begun to realize that we can understand the mind in the same way that we understand computers.

This way of looking at the mind is called computer metaphor(Newell & Simon, 1972). Although this metaphor is well-known among researchers and even some researchers call it the cornerstone of cognitive science research, we believe that most researchers' understanding of this metaphor is only at the perceptual stage, and its deeper implications are overlooked.

Just imagine, if the mind is working like computers, what kind of computer is it? This question is not difficult to answer. If we want a computer to be able to handle all kinds of problems, it needs to play at least two roles: 1) The computer itself; 2) the Programmer. Programmers make the computer can deal with various problems by writing corresponding programs. If the mind is working like a computer, then it must play both roles. In other words, it must be a computer system that can program itself, which we will refer to as a self-programming system hereafter.

If we accept the above idea, it means that there are many different subprograms in the mind. These subprograms are coded by the mind itself for dealing with various tasks in different environments.

If we study such a computing system by the method of studying the objective world—deriving laws from phenomena—there will be a problem. This problem is that we will get many different theories and each theory only captures the laws expressed by the mind-created subprograms other than the mechanism of the mind itself. From a practical point of view, these laws seem to explain some but not all phenomena. Sometimes certain laws work at the same time, but in other times they can contradict each other.

The situation described above is exactly the current state of the research of the mind. There are several independent explanations for almost every phenomenon of the mind. Each of these explanations can only work on some of these phenomena but failed on others. For example, if we ask how does the mind do inductive reasoning? So the answer given by

the current research is like this: Some of the evidence shows that the basis of inductive reasoning is the similarity between categories(López et al., 1997; Osherson et al., 1990; Rips, 1975), namely-- if objects belonging to one category have a certain property, another category similar to this category may also have that property; Other evidence points out that inductive reasoning is not merely dependent on the similarity between categories but also dependent on properties themselves. That is, if a property is widely possessed by lots of categories, it is more likely to be present in a new, unknown category(Nisbett, 1983). Meanwhile, another part of scholars believes that the mind makes inductive reasoning according to the scientific categorization(Carey, 1985; Gopnik & Meltzoff 1997; Keil, 1989; Murphy & Medin, 1985).

Even if we don't ask where "categories" in these answers come from and even combine these answers above together, it is still far from a complete one. For example, a directly following up question is, if inductive reasoning is based on the combination of the mechanism summarized by the above answers, how do these mechanisms interact with each other?

Situations like this are not unique in the research of the mind, and indeed if we traverse all fields of cognitive science we will find similar phenomena everywhere (Holyoak & Morrison, 2005; Richters, 2021).

So why do there exist such phenomena in the research of the mind? In fact, if the mind works as a computing system that can program itself and we study it by the paradigm of studying the objective world, then these phenomena will be no more than an inevitable result. This is because these answers obtained by such a paradigm do not grasp the mechanism of the mind itself, but only discovered the effective thinking mode generated by the mind according to the past interaction with the external world.

So how can we figure out the mechanism of the mind without falling into the trap above? The correct starting point is to go back to rethink the doctrine that the mind works as a self-programming system. This doctrine points out two key criteria of this computing system: 1) The system should be able to represent any laws that can be summarized by people and represent any programs that can utilize these laws; 2) This system can generate sub-program and apply them to adapt to the external world.

If we only require a computing system to meet the first criterion, then there is already

an answer, that is, a Turing machine or a general-purpose computer from the same vein. But it is extremely difficult for such computers to meet the second criterion — self-programming for adapting to the environment. This difficulty lies in the fact that the Turing machine itself is very simple in its storage structure. Its memory is just a paper tape and its addressing method only includes sequential execution and jumping. Trying to produce a system that can self-programming based on such a simple storage structure is as difficult as asking a monkey to write the book Hamlet by hitting a random keyboard.

The way to solve this problem is to give the computing system a more complex and meaningful storage structure, and then a bootstrap program can use this storage structure to reprogram itself. In this sense, the storage structure corresponds to the hardware of a computer, and the bootstrap program corresponds to the operating system running on it. The goal of this paper is to solve the first problem, which is to propose a suitable storage structure. And the bootstrap program will be discussed in our next paper.

However, before discussing the storage structure in detail, we must first discuss another key issue - the problem of inductive reasoning. Because inductive reasoning is the core function that the storage structure must support, only if this problem is well understood, the meaning of the storage structure can turn out to be clear. Moreover, by solving this problem, we can see how a self-programming computing system can bring fresh new and important insights to us.

2. The Problem of Inductive Reasoning

The problem of inductive reasoning originated from the philosophical discussion of the source of truth. Hume pointed out that one cannot find a truth by inductive reasoning(Hume, 1739, 1784). There are many examples supporting Hume's point of view, one of the most famous examples is that of Russell's Chicken.

There is a flock of chickens on the farm. Among these chickens, one of them is an inductivist. it discovered that: Every morning, food appeared in the trough. After a long period of observation, it found that this phenomenon is bound to happen no matter what the circumstances. So it's ready to release this big discovery at Christmas. But on Christmas Day, it was shocked to find that there was no food in the trough, but a butcher with a knife. Then, the chicken inductivist became the meal in the evening.

As we can see from this example, no matter how many times an event is repeated, it doesn't mean that the event will always be like this in the future. This is the limitation of the inductive reasoning proposed by Hume. This limitation leads to a more practical question: If induction fails to find an eternal truth, what are our predictions based on? An alternative idea is to accept probabilities. Russell summed up such process of inductive reasoning: if two events have never happened alone, the more times these two events happen together in the past, the greater the chance that these two events will happen together in the future(Russell, 1959). But no matter how many times these two events happen together in the past， the chance in the future is not 100%.

But is it really that simple? Just imagine such a situation, if the sun hasn't risen today, what is the probability that the sun will not rise tomorrow? Then consider again if you won a lottery ticket with a 1/10,000 winning chance today, what will be the chance of winning tomorrow? I think the vast majority of people would find them very different, although, from the point of view of observation, the number of times we have observed the sun rising is about tens of thousands, and so the probability of not rising in the next day should be also about one in tens of thousands. But if the sun doesn't rise today, we're likely to think that it probably won't rise tomorrow, because maybe there's something big has happened. But we never think like that about lottery tickets.

So how does the mind solve prediction problems? First of all, it is irrefutable that the prediction is to speculate on future phenomena through past phenomena. If we think about this problem from the perspective of the self-programming system, we can think of the past phenomenon and the future phenomenon as two completely different storage objects in the mind. What connects the two is a sub-program—a sub-program that infers the future from the past.

Up to this point, the views of the predecessors are consistent with ours. And the key disagreement is in the next part - what is this sub-program?

No matter Russell's Chicken uses past events to directly predict events in the future or uses probability to describe the chance of an event will happen in the future, they are all described some particular sub-programs that can be used to predict the future from the past. Both of the two sub-programs have one characteristic is that they are universal. That is, the way the sub-program works is independent of a particular event. Take our previous

example, the way to calculate the sun will not rise and the chance of winning a lottery is the same. To search for such a universal sub-program was the goal of past philosophers and cognitive scientists.

However, such a universal sub-program is neither necessary nor optimal for a self-programming system. From the perspective of necessity, the system does not require such a universal sub-program. It can write such sub-programs when it is needed. From the perspective of optimality, using different sub-programs to handle predictions within different domains will generally be better than the predictions from a universal sub-program. For example, in the example that the sun did not rise today, we are likely to consider the underlying physical principles, and then make conjectures based on this principle. Such predictions are much more reliable than intuitive probabilistic predictions.

In fact, predictions like these that do not base on direct observation are ubiquitous in our human activities. For example, our scientists can predict that the sun will vanish after around ten billions year.

To sum up, our view of the problem of inductive reasoning is that past facts and events in the future are connected by a variety of different sub-programs, rather than existing a universal sub-program as philosophers and cognitive scientists believed in the past. But this explanation alone is obviously not enough. For a universal single program perhaps we can attribute it to biological evolution, whereas for a number of different programs we obviously have to explain where they are come from.

For this problem, we believe that these sub-programs are still generated by inductive reasoning. But at this time the goal for inductive reasoning is no longer to find out the relationship between events but to find out the effective sub-program for prediction in a particular environment.

Here, some readers may point out that such an explanation fails to solve the problem of inductive reasoning. Because the so-called effectiveness of a sub-program is also obtained by inductive reasoning, doesn't it still require a universal program ultimately?

Our answer to this question is that the first half is correct, but we disagree with the second half. Our reason is that since the sub-program for inductive reasoning can act on other sub-program for inductive reasoning, there only needs an initial sub-program for inductive reasoning. This initial sub-program can generate more complex sub-programs

for inductive reasoning. These newly created sub-programs can gradually replace the initial sub-program in a variety of specific environments.

More generally, the above method of using induction to discover new inductive sub-program can also be used to discover sub-programs to establish other relationships. The importance of this is that if the information in the mind is organized in some relationships, then induction is sufficient to support the discovery of such relationships. One of the very important relationships is the causality that we will explain in later chapters.

Compared to the belief of existing a universal induction sub-program, the method of starting from an initial induction sub-program and then gradually evolution later is more adaptable. But at the same time, it puts forward requirements on the storage structure of information. Specifically, if a storage structure can support this gradual evolution, it needs to support: 1)the sub-programs of inductions can be performed based on it. 2) The source code of these sub-programs can be recorded in it. The former requirement indicates that inductive reasoning can operate rely on past information, and the latter indicates that the effective inductive reasoning method in the past can be recorded in the storage structure. Only with such a storage structure can the self-programming system really run.

With the above understanding of induction from the perspective of the self-programming system, we can now return to the original philosophical question: what is the relationship between induction and truth? We believe that since the root of our mind in understanding the world is still induction, we agree with Hume: the mind cannot obtain absolute truth. However, such an answer is not enough, since it will lead to a follow-up question: if obtaining truth is not a reason for the justification of induction, then what is the source of its justification while induction is often the main approach for reasoning in our daily lives? In fact, this problem is simple under the self-programming system. Because one of the main tasks of the self-programming system is prediction, it automatically collects information that can predict the future. In other words, information without predictive power is likely to be excluded at the information gathering and organizing stage. Therefore, when people apply induction, they have actually ruled out those repeating phenomena, however, without predictive power.

Although, from the perspective of self-programming, it is not difficult to draw conclusions about the problem of induction, the problem of induction, as one of the most

important and widely discussed problems in epistemology, it is necessary for us to compare our conclusions with other past theories for a more comprehensive understanding.

Generally speaking, past theories about the problem of induction can be divided into two categories, namely, the justification of induction can or cannot be achieved (Henderson, 2020; Climenhaga, 2021).

In the first category which argues that induction is provable, there are two schools of thought. The first school believes that we can model the world and then equate induction with some kind of rationality in the model. In this way, induction could be justified in the current world. For example, the theory of inference to the best explanation (IBE, Bonjour, 1997; Foster, 2004), or more recently the meta-induction theory (Schurz, 2019).

Although each of these theories faces critics respectively, there are some common problems that are overlooked in past literature. First, the problem of induction occurs at a very basic level which is identifying an object. The essence of identifying is to find a property that will not change at least until the next observation, that is, to infer future properties from the properties of the current object. This means, that if we agree that there is an identity of objects in a model, we have in fact assumed the justification of the problem of induction. For example, in meta-induction theory, there are multiple players that can be observed. This actually assumed the existence of the identity of these players and such assumption implicitly assumed the justification of induction.

In addition, and more importantly, the justification of the process of modeling the world actually relies on the justification of induction too. This is because modeling is actually a way of looking at the world by excluding unimportant factors under the current problem and extracting important factors. So what should be excluded and what should be put in the model is actually summed up by past experience. Therefore, assuming that the current modeling is sound is equivalent to assuming that induction has been justified. Therefore, we believe that the first school to justify induction is not actually feasible.

Next, let's look at the second school. The two main viewpoints of the second school are that circular arguments are either considered benign, or circular arguments can ultimately end by some self-evident assertion, such as the natural kind theory (Sankey, 1997; Lange, 2004 & 2011) or the material theory of induction (Norton, 2003 & 2014). For the former, the core problem they face is that counter-induction can also be circular, but it

contradicts the result of induction. As for the latter, apart from induction, we cannot actually find any assertions in the empirical world that are self-evident. In this dilemma, none of these explanations are convincing enough (Okasha, 2005; Kelly, 2010; Worrall, 2010; Peden, 2019; Bassford, 2022).

Finally, let's look at the second category which believes that induction is not justifiable. First, our theories also fall into this category. However, although other theories in this category, such as aim-oriented empiricism (Maxwell, 1993 & 2017), hold the same conclusion as us, they fail to distinguish the induction itself and the induction that is perceived. In our theory, such distinction and the interpretation of the latter are at the heart of how we solve the problem of induction.

In fact, notwithstanding their ultimate point of view, many theories in the past have more or less captured some aspect of induction. For example, meta-induction theory argues that induction occurs at the meta-level, which is consistent with the interpretation from the self-programming systems, but we have a different view of what meta-level is based on.

Or Sober's background theory (1988) and Norton's material theory agree with us on whether there is a uniform algorithm for induction. But, Sober does not further explain where the validity of the perceived induction comes from. Norton's theory differs from ours in terms of what the induction algorithm depends on. So we finally got the opposite conclusion.

Or the phenomena described by natural kind theory and aim-oriented empiricism are actually the results generated by the self-programming based on the goals they need to achieve. In other words, they can also each be regarded as special cases of self-programming systems in different aspects.

Even Kant's viewpoint on the law of nature is consistent with the source of the validity of the inductive mentioned here. The difference between these two is that a lot of knowledge that Kant deemed *a priori* can be acquired by the self-programming system *a posterieri*.

3. Storage Structure

Next, we will return to our original the question: what kind of storage structure can support the self-programming system? First, we need to re-emphasize that our goal is not

to propose a new storage structure to solve problems beyond the computational ability of general-purpose computers, but rather a more convenient storage structure that makes self-programming less difficult. The decrease of such difficulty is like converting the task of randomly creating a house with lime and bricks to the task of creating a house by randomly combining roofs, walls, and foundations. Based on this purpose, we can deduce what kind of storage structure this self-programming system should have from the basic functions that need to be implemented.

3.1 The basic functions of the self-programming system

We believe that the basic functions of self-programming systems should include at least two points: 1) Trigger corresponding behaviors according to both external and internal environment; 2) Modify behaviors according to past feedback on behaviors. Both of these functions are unquestionably necessary for the survival of individuals in a complex world. And they are not only applied to humans but also to animals.

Next, let's take a closer look at these two functions. The first one —the trigger of behavior — can be further divided into two steps: First, locate the specific location in the storage structure according to the conditions from the environment; Second, find and execute the corresponding operations according to this specific location in the storage structure and the demand of the mind.

If we try to analogy the above steps to an existing function from a general-purpose computer, it is equivalent to finding the entry point of a program through conditions and results. And this is not a basic function of a general-purpose computer. In a general-purpose computer, the role of the computer is to calculate result when the conditions and the program that was used to calculate is known. In this process, the entry point of the program is given by the programmer, so the computer itself does not need to consider this problem. But in the self-programming system, because it must play the dual roles of programmer and computer at the same time, it is essential to locate the program entry, so how to realize this function is one of the core issues of the self-programming system.

Let's move on to the second basic function - modifying behavior based on negative feedback. Specifically, this negative feedback refers to situations where the expected event does not happen as expected. What is the meaning of revising expectations in this

situation? As discussed in the previous chapter, the past facts and the expected events are connected by a sub-program. So the revisions of expectations are actually revisions to the method to make expectations, that is, the sub-program that made the wrong expectations. This revision requires the storage structure can support locating and modifying the source code of these sub-programs.

If we want such a function in the system, we need its storage structure that allows the following three operations: 1) Running the sub-program for expectation and store its result; 2) Tracking the error; 3) Use the tracked error to locate the sub-program that make expectation.

According to the required functions above, we can see that, no matter triggering behavior or revision behavior, a necessary step is locating the sub-program. However, the located sub-program are used in different ways in these two functions. The former treats it as executable code, while the latter treats it as a data object that can be modified.

A system with a storage structure that can play these dual roles has endless potential. Because it implies that a program can modify and generate new source code. And the program represented by the generated source code can generate more source code again. This iteration can run forever for creating the most adaptive behavior.

This powerful capability requires the storage structure represent the source code and the data in the same form. Just like in a general-purpose computer, both code and data exist in the form of text. This allows programs that process text to act on both code and data. In the case of a self-programming system, we also need a storage structure that can fulfill the requirement.

3.2 The form of the storage structure

So what should a storage structure that not only satisfies the above basic function but also facilitate self-programming look like? While a general-purpose computer-like approach to using text to store code theoretically allows for self-programming, it is not the best option from a practical perspective. The reason is that it is still too complicated to generate a usable sub-program in its linear memory format.

Our answer to the above question is that all information in a self-programming system should be stored in the form of sets of properties. If we only answer this far, the idea is not

new and useful at all, and it can even be considered a very outdated idea. As early as thousands of years ago, Aristotle has put forward the idea that the concept in the human mind is a set of properties. But it will have a whole new meaning if we go a step further to understand and define more details based on the two basic functions we've proposed earlier.

First, what does the property refer to in the above definition ? Generally speaking, the commonsense definition of the property is the perception that we can directly experience, such as color, shape, or smell. But the properties we mentioned here have a much more general meaning.

Specifically, to any object, if we place any specific operations on it, it will turn out to have some specific result, it can be defined as a property of this object. The result here refers to another property set or some inherently recognizable objects, such as a certain smell, color, shape, etc. Under this definition, a property can be defined no matter how complex the related operation is. For example, we can conduct a complex physical experiment to measure the spin of a microscopic particle, no matter how complex the experiment is. This process and the final result together constitute one property of this particle. By this way of defining, we can also express naive properties like color. For example, the property that an object is red is defined by the operation of "seeing" and the triggered internal representation of color in the mind.

As mentioned above, properties are defined by operations, but what are operations? In the past, scholars usually thought of operations as visible and tangible actions, but here we believe that mental actions are equally or even more important. What are these mental actions? We believe that the most important mental action are these operations of manipulating the storage system consisting of the property sets. These operations include creating a new property set, comparing a property set with others, modifying a property in a property set, performing an operation in a property, and so on.

Next, we use the following example to demonstrate the utility of the property sets. First, we start with a very simple one. A property set represents a concrete object -- an apple. This set has properties as the following: Property 1. Smells like an apple; Property 2. Looks like an apple; Property 3. It can be eaten for eliminating hunger.

At this time, suppose there is a baby, and the above property set exists in her storage

system. At the same time, there is no other property set in her storage system that has Property 1 but does not have Property 2 and Property 3. Let us again assume that the infant's initial judgment method is as Rule 1 - if an external object can trigger a certain property set, then it must have all the other properties belonging to this property set.

In this case, if the baby happens to be hungry and sees an apple at the same time. She will deduce from Rule 1 that eating it will definitely eliminate hunger. This is the most basic application of the property sets. Next, let's make the situation a little more complicated.

One day, her mother gave the baby a fake apple, which has Property 1 but does not have Property 2 and 3. So what will happen to the baby's storage system? First she will build a new property set with property 1 but without Property 2 and Property 3 to represent the fake apple. But that's not enough, she also needs a new rule for what to do when an observation triggers both real and fake apples at the same time.

In this case, several different rules are possible:

Rule 2 - If an object triggers both a real and a fake apple, I should smell it again to see if Property 2 is met, then to make whether to eat it.

Rule 3 - If an object triggers multiple property sets at the same time, it is necessary to continue to collect more properties to ensure that all property sets that are not excluded have the Property that represents "can be eaten".

Rule 4 - If an object triggers multiple property sets at the same time, it is necessary to continue collecting more properties to ensure that all property sets that are not excluded have the property which is needed.

Rule 5 (Hume's Rule) -- Even if the observation excludes all exceptional property sets that don't have the property which is needed, it is not certain that the object can satisfy the demand.

We can see that the above Rule 2, 3, and 4 can all meet the requirements of the application. But the abstract levels they express are different. Rule 2 can only apply to apples and eating. Rule 3 discards the condition of being an apple and converts it to the environmental conditions of the property set in the storage system, but still retains the property of eating. Rule 4 is a purely abstract rule. It has nothing to do with concrete things. In other words, it can not only solve the problem of eating, but also the problem of any aim,

and even the problem of method choice. Rule 5, on the other hand, is a rule that considers not only the current storage system but also the past state of the storage system.

For the above example, the first point to note is that all these rules can actually be expressed in the form of property sets because the conditions and operations expressed by these rules are just the compositions of the queries and comparisons of the storage system.

Secondly, what we call abstract or concrete in commonsense does not have an essential distinction. They differ only in whether the operations that make up the properties belong to the body or to the storage system. In this case, the process of building categories is just a process of intersecting property sets.

Finally, some readers may think that hybrid property sets like Rule 3 are useless. But we speculate that this type of property set is probably numerous since it provides a rule for gathering information based on concrete conditions and already known property sets in the storage system. We think this type of set is very applicable to problems like visual recognition. By using the property set that expresses such rules, we can make the recognition of everything in the world start with a simple color block, and then further extract more properties according to the requirements from the storage system. The idea here is consistent with the idea of Active Vision. We believe that only the use of self-programming systems can truly solve the algorithmic problem of active vision.

In summary, solving the problem of storage structure has two significances. From a practical point of view, it provides an execution environment for the self-programming systems. In other words, all other functional modules can be regarded as things like the peripherals of a computer. They are incorporated into the execution environment by interrupting and providing functionality. Besides the body, these peripherals may also have other modules. For example, a module that combines operations can help to form new operations and properties.

In addition, from a philosophical point of view, since the abstract operation in thinking can be interpreted as a conditioned reflex of the status of property sets and operations of storage structure, we can think that the operation of thinking is essentially an ongoing chain of conditioned reflexes.

3.3 Storage structure and deductive reasoning

After understanding the mechanism of property sets to make predictions, we can return to the question we proposed before: what deductive reasoning is. Deductive reasoning is a reasoning process from general to special, and the form of its reasoning is summarized by Aristotle as a syllogism. In simple terms, syllogism contains a major premise, minor premise, and conclusion. For example, the major premise: All students in the economics department are qualified in mathematics. The minor premise is that A is a student of the Department of Economics. Then the conclusion can be that the mathematics of A is qualified.

Just imagine, how can we use the storage structure discussed above to understand this example? The first step is to represent the major premise - that all students in the Department of Economics are qualified in mathematics. Using the language of property sets, this major premise means that there is a property set for prediction which includes two properties. Property 1 indicates that an object is a student belonging to the Department of Economics; Property 2 indicates that an object's mathematics is qualified. When a specific student from the Department of Economics appears, that is, when the minor premise appears, the mind can use this property set to infer that the specific student is qualified in mathematics. This will form a new property set for this student. This property set contains additional specific properties beyond these two properties in the set of the major premise, such as the student's height, appearance, and so on. The generation of the property set that represents the specific students is exactly the function of the property set of the major premise. In other words, the reason why the mind has to form this kind of property set for prediction is that the mind needs this general-to-special process to be correct.

From this point of view, what deductive reasoning represents is the function of the storage structure, whereas induction reasoning is the root to generate such storage structure. If we analogize a property set to a car, then inductive reasoning is equivalent to the method used to make the car, while deductive reasoning expresses how the car can be used.

To sum up, we can get a new perspective on deduction: the reason why deduction is always correct is not that it is truth as European rationalists thought it was. Rather, it is because in the mind the process of deductive reasoning is exactly the process of how to use the storage structure in the mind. Since the correctness of this process is determined by the definition of the property set, it will always be true.

3.4 Storage structure and causality

Finally, let's re-examine what the property sets mean from another perspective. In fact, a property set expresses two kinds of relationships. The first is the relationship that how to change an object from one state to another state. This relationship is represented by a property. The other one is the relationship between properties. We believe that these two relationships are exactly the essence of causality and correlation respectively. The latter is easy to understand. We can easily use property sets to represent correlations. If a tomato is very red, we can deduce that it should be delicious. Very red and delicious can be viewed as two properties belonging to the set of tomatoes. In such a way, the correlation between the color red and the delicious taste can be expressed. But why is the former relationship causality? This needs us to start with the long-standing puzzle of causality.

The 18th-century British philosopher David Hume questioned whether the law of causality was real. In Hume's view, causality cannot be directly observed, so he believes that causality is just an illusion in the mind. Take the following example "the branch of the apple tree breaks and the apple falls". these two events that we can observe are "the branch breaks" and "the apple falls". But whether there is a causal relationship between the two events is unobservable. From this, Hume further concluded that causality is nothing but a name for the empirical correlation that is always adjacent in time and space. This view of causality proposed by Hume is called "the constant conjunction theory".

But can this theory explain causality? The answer is no. For example, crows of roosters always occur when the sun is about to rise, but we obviously don't think of roosters as the reason for the sun's rise. This shows that the events that happened one after the other in time may not have a causal relationship.

To look at another example, if you find your alarm doesn't go off one morning, it's most likely because you forgot to set your alarm last night. In this case, there is no temporal adjacency between these two events "the alarm didn't go off" and "forgot to set the alarm".

From the above two examples, we can see that Hume's constant conjunction theory is neither a sufficient nor a necessary condition of causality. In fact, Hume himself was not unaware of the problems of this theory. He once noted:

"We may define a cause to be an object followed by another, and where all the objects,

similar to the first, are followed by objects similar to the second. Or, in other words, where, if the first object had not been, the second never had existed."

However, later researchers pointed out that Hume's statement actually pointed to two different theories, not simply "in other words". Because the second half of its statement — if the first object had not been, the second never had existed— points out a whole new definition of causality. This definition is called counterfactual theory by later researchers(Pinker, 2007).

While scholars generally agree that counterfactual theory is a more tenable theory than constant conjunction theory, critics quickly pointed out that there are fundamental problems with the counterfactual theory. Critics argue that we cannot replay our lives like movie replays, so it is impossible for us to observe another situation in a world that did not happen. So, for a counterfactual theory to hold, the key task is to explain what this unobserved possible world is.

In our theory, this unobservable possible world is exactly the property set. Recall that properties in a property set represent a combination of operation and outcome. This is equivalent to say that a sufficient condition for a combination of operation and outcome to hold is that they belong to this property set. In other words, the property set actually represents the environment in which such a combination can hold. This environment also contains combinations of other operations and outcomes. As an example, take a property set representing a vase on a table. It contains Property A which is "hit it with a hammer and it will break" and another Property B which is "leave it there and it will stay fine". If the event represented by Property A occurred, then there is a counterfactual contrast to the event represented by Property B that did not occur. Thus the behavior "hitting with a hammer" in Property A becomes the reason for the broken vase.

To sum up, the unobservable possible world of counterfactual theory actually exists, but it exists in the storage structure of the mind, not in the real world. Kant once pointed out that causality is only an innate form of organizing experience(Kant & Smith, 1929). In this sense, the property set we point out is the implementation of this innate form. And we can go a step further and think that not only causality, but even correlations are also innate forms. These two relationships together construct the information organizing principle of a self-programming system like the human mind.

4. Summary

How the mind works have always been the most fascinating question. However, the methods of the past have always looked for a specific law to explain a specific phenomenon generated from the mind. We think this way is unable to really find out the nature of the mind. Laws found in this way are, at best, the description of subprograms created by the mind.

The correct approach should be to think of the mind as a computing system that can code itself. This system, like our general-purpose computer, has its own storage structure, basic instructions, and basic symbols. What differentiates it from a general-purpose computer is its ability to form more complex programs on its own using a simple set of bootstrap programs. Furthermore, these programs can substitute the bootstrap program and guide how the mind works.

As the first step to realizing this idea, we proposed a storage structure that can meet the requirements of self-programming, which is the initial bedstone of solving the problem of self-programming. We will also rely on this storage structure in subsequent papers to explain the implementation of the second component of the self-programming system - the bootstrap program.

Reference

Bassford, Andrew Dennis (2022). An Intuitive Solution to the Problem of Induction. *Principia: An International Journal of Epistemology*.

BonJour, Laurence (1997). *In Defense of Pure Reason: A Rationalist Account of a Priori Justification*. Cambridge, England: Cambridge University Press.

Carey, S. (1985). *Conceptual change in childhood*. The MIT Press.

Climenhaga, Nevin (2021). Evidence and Inductive Inference. In Maria Lasonen-Aarnio & Clayton Littlejohn (eds.), *The Routledge Handbook of the Philosophy of Evidence*. Routledge.

Foster, John (2004). *Divine Lawmaker*. Oxford University Press UK.

Gopnik, A., & Meltzoff, A. N. (1997). *Words, thoughts, and theories.* The MIT Press.

Henderson, Leah, "The Problem of Induction", *The Stanford Encyclopedia of Philosophy* (Spring 2020 Edition), Edward N. Zalta (ed.), URL = <https://plato.stanford.edu/archives/spr2020/entries/induction-problem/>.

Holyoak, K. J., & Morrison, R. G. (Eds.). (2005). *The Cambridge handbook of thinking and reasoning.* Cambridge University Press.

Hume, D. (1739). *A treatise of human nature: Being an attempt to introduce the experimental method of reasoning into moral subjects. London*: Printed for J. Noon

Hume, D. (1748). An inquiry concerning human understanding. Oxford, UK: Clarendon

Kant, I., & Smith, N. K. (1929). *Immanuel Kant's Critique of pure reason*. Boston: Bedford.

Keil, F. C. (1989). *Concepts, kinds, and cognitive development.* The MIT Press.

Kelly, Thomas (2010). Hume, Norton, and Induction without Rules. *Philosophy of Science* 77 (5):754-764.

Lange, Marc (2004). Would "direct realism" resolve the classical problem of induction? *Noûs* 38 (2):197–232.

Lange, M. (2011). Hume and the Problem of Induction. Edited by D. M. Gabbay, S. Hartmann, & J. Woods, *Inductive Logic* (43–91). doi:10.1016/B978-0-444-52936-7.50002-1

López, A., Atran, S., Coley, J. D., Medin, D. L., & Smith, E. E. (1997). The tree of life: Universal and cultural features of folkbiological taxonomies and inductions. *Cognitive Psychology*, 32(3), 251–295. https://doi.org/10.1006/cogp.1997.0651

Maxwell, Nicholas (1993). Induction and scientific realism: Einstein versus Van Fraassen part one: How to solve the problem of induction. *British Journal for the Philosophy of Science* 44 (1):61-79.

Maxwell, Nicholas (2017). *Understanding Scientific Progress: Aim-Oriented Empiricism*. St. Paul, USA: Paragon House.

Murphy, G. L., & Medin, D. L. (1985). The role of theories in conceptual coherence. *Psychological Review, 92*(3), 289–316. https://doi.org/10.1037/0033-295X.92.3.289

Newell, A., & Simon, H. A. (1972). *Human problem solving*. Prentice-Hall.

Nisbett, R. E., Krantz, D. H., Jepson, C., & Kunda, Z. (1983). The use of statistical heuristics in everyday inductive reasoning. *Psychological Review, 90*(4), 339–363. https://doi.org/10.1037/0033-295X.90.4.339

Norton, John D. (2003). A material theory of induction. *Philosophy of Science* 70 (4):647-670.

Norton, John D. (2014). A material dissolution of the problem of induction. *Synthese* 191 (4):1-20.

Okasha, Samir (2005). Does Hume's argument against induction rest on a quantifier-shift fallacy? *Proceedings of the Aristotelian Society* 105 (2):253-271.

Osherson, D. N., Smith, E. E., Wilkie, O., López, A., & Shafir, E. (1990). Category-based induction. *Psychological Review, 97*(2), 185–200. https://doi.org/10.1037/0033-295X.97.2.185

Peden, William (2019). Direct Inference in the Material Theory of Induction. *Philosophy of Science* 86 (4):672-695.

Pinker, S. (2007). *The stuff of thought: Language as a window into human nature.* Viking.

Richters, J. E. (2021). Incredible utility: The lost causes and causal debris of psychological science. *Basic and Applied Social Psychology, 43*(6), 366–405. https://doi.org/10.1080/01973533.2021.1979003

Rips, L. J. (1975). Inductive judgments about natural categories. *Journal of Verbal Learning & Verbal Behavior, 14*(6), 665–681. https://doi.org/10.1016/S0022-5371(75)80055-7

Russell, B. (1959). *The problems of philosophy.* New York: Oxford University Press.

Sankey, Howard (1997). Induction and Natural Kinds. *Principia: An International Journal of Epistemology* 1 (2):239-254.

Schurz, G. (2019). *Hume's Problem Solved.* MIT Press.

Sober, Elliott (1988). *Reconstructing The Past: Parsimony, Evolution, and Inference*. MIT Press.

Worrall, John (2010). For Universal Rules, Against Induction. *Philosophy of Science* 77 (5):740-753.