A Runtime Framework of the Mind

Fangfang Li

Galileo Mind Research

Liff1229@hotmail.com

Xiaojie Zhang

Chongqing Medical
and Pharmaceutical College
10802@cqmpc.edu.cn

Abstract

How the mind works is one of the most fascinating unsolved mysteries for human beings. Throughout history, countless scholars have tried to answer this ultimate question. But none of these answers satisfy the vast majority of scholars like a law of physics. This paper proposes a core principle by which the mind operates. We call it the self-programming system. The self-programming system can learn, store and apply the functions of bodies, external tools, and even the mind itself in the same way. We prospect this system can be developed to truly solve this ultimate question. However, due to the generality of the mind, the traditional scientific methods of verifying theories, no matter theoretical or empirical, are not suitable for verifying a true theory of mind. Therefore, we have to appeal to show the explanatory power of the self-programming system. In this paper, we, first, compare it with the past self-improving AI frameworks to show their weaknesses and how these weaknesses are overcome. Then, by comparing with mainstream neuroscience-based theories of consciousness, we found each of these theories captures a different aspect of consciousness. The theory of consciousness based on self-programming systems plays all their roles. This result implies the empirical data that supports these mainstream consciousness theories can also indirectly support our consciousness theory. Combining the analysis of induction and causation in our

previous article, we have demonstrated the explanatory power of the self-programming systems from four aspects.

Keywords: Self-improving AI, Consciousness, Human-level AI, Epistemology，Embodied cognition

1. Introduction

How the mind works is one of the most fascinating unsolved mysteries for human beings. In our previous article, we pointed out that the mind should be viewed as a self-programming system. (Li, 2022) Then, in order to build this self-programming system, we proposed a storage system for storing knowledge. Based on this storage system, we explained the relationship between inductive and deductive reasoning and also concluded the nature of causality.

In this article, we will further discuss how this storage system can support the runtime of the self-programming system, specifically, how the self-programming system can run based on the storage system and how to learn new knowledge. Through an analysis of the runtime and learning mechanisms of the self-programming system, we will show how the system achieves self-improvement, which has long been considered a key feature of human-level artificial intelligence. At the same time, we will explain what consciousness is and why it is necessary to this system.

But before proceeding in detail, it is necessary for us to discuss the idea and purpose of our paper from the perspective of the methodology of verifying scientific theories. The reason we do this seemingly superfluous step is that there is an irreconcilable contradiction between the nature of the mind and the current scientific research paradigm. Therefore, we have to choose a way to validate our theory that does not quite fit the current scientific research paradigm. Due to this situation, if we don't clarify these contradictions, it may lead readers to misinterpret the relationship between the problem we discussed and our proposed framework.

First, let's look at why the traditional paradigm of verification theory is not applicable to validate the theory of mind. In the current scientific paradigm, a theory is justified either through formal methods or experimental data. Let's begin with the justification by formal methods.

Specifically, to formally validate a theory of mind, what we actually do is to assume that the behavior of the agent conforms to certain rules, and then assume that the world

in which the agent exists has certain laws that are independent of the behavior of the agent. Then we can justify the theory by showing that the agent's behavior is reasonable/optimal/rational under the assumption about the world.

But, such reasoning is not reliable for validating a theory of mind. The key reason is that assuming the laws of the world is equivalent to admitting these laws are the absolute truths of a world. So the effectiveness of this proof must appeal to there exists an absolute truth and it has been attained and properly expressed in the assumption.

However, both from our experience of daily lives and scientific explorations, we know a key characteristic of our mind is that we can only progressively find more effective and general relative truths rather than relying on assumptions of absolute ones. Thus, we argue that once a theory of mind assumes some laws of the world independent of the observer will inevitably lose its generality, since it is only applicable to a special world following some particular laws. However, if we don't make assumptions about the world, but only assumptions about the mechanism of the mind, then formal methods will not lead to any meaningful results.

Now, let's look at verifying theories through empirical data. For a theory of mind or human-level intelligence, such methods can be further divided into two cases. The first is to only verify in a specific domain. The second is to verify by building a real concrete agent and placing it in the real world. And then examining its performance by the standard for a normal human.

The first is actually the current method of verifying weak artificial intelligence. However, this verification method is meaningless for human-level intelligence.

Regardless of whether an artificial agent performs below or above humans in a special domain, it cannot represent the correctness or even the incorrectness of the theory. This is because, if an agent is inferior to humans, it can be attributed to the knowledge that humans have learned in other domains being used in this specific domain. The reason why humans are smarter is not because of the incorrectness of the theory. The reason could be the knowledge from this specific domain is not self-contain. In other words, to result that the theory is wrong from the fact that the agent underperforms humans must rely on the epistemic independence of each domain.

However, such independence is false even from the everyday experience of ordinary lives. Even for mathematics, which is usually considered domain-independent, this is still not the case. Because it is actually independent only in knowledge, but not in the way of what and how mathematical knowledge is inspired and accumulated.

On the other hand, outperforming humans in a specific domain does not imply the correctness of a theory of mind. For example, in doing arithmetic, computer programs are far superior to humans, but we do not think that arithmetic programs are a program of the mind.

The second way of verification empirically is to put an agent completely into human society, raise it like a child, and then verify its effectiveness. This method is theoretically feasible, but its main drawback is that there will be no stage results to help further exploration. Therefore, this approach, even if feasible, does not fit the path of the scientific community to explore the world.

The second way of verification empirically is to put an agent completely into human society, raise it like a child, and then verify its effectiveness. This method is theoretically feasible, but its main drawback is that there will be no stage results to help further exploration. Therefore, this approach, even if feasible, does not fit the path of the scientific community to explore the world.

In fact, the above discussion of methodology for validating theories of human-like intelligence is far from our own. For example, the reason Turing invented the subjective-oriented Turing test as a way of verifying artificial intelligence could result from the similar reasons that we discussed above. (Turing, 1950)

So is there any way to perform phased verification? The answer is yes, but this approach is not quite fit with current scientific paradigms and more akin to philosophical speculative processes. Philosopher Dreyfus (1972) has pointed out that a correct mental model should be able to bring insightful answers to many questions in epistemology and philosophy of mind. By following this way, we argue that the theory of mind can be validated by examining the explanatory power of the theory. It's like Newton's laws' theoretical power comes from explaining various phenomena within a

single framework. The correct theory of mind should be able to explain all kinds of thinking phenomena under a single framework.

And that's the approach we're currently using. What we hope is that we can solve every problem of epistemology and cognitive tasks by applying our self-programming framework. However, since every epistemological problem is far more complex than general physical problems, our goal can only be achieved step by step. In our last article, we used the self-programming framework to explain Hume's problem of induction, the nature of deduction, and the nature of causality. In this article, we mainly deal with consciousness.

Under our framework, understanding conscious and unconscious thought processes is a prerequisite for solving all other cognitive tasks. In fact, even if we only consider our daily life experiences, we can find that cognitive tasks like, recognizing objects, problem-solving, decision-making, or using language, have both conscious and unconscious processes. Therefore, any general theory prospect to solving how the mind works must first solve the problem of consciousness. In our case, we need to explain in future articles how to use self-programming systems instead of connectionist methods to solve recognition problems based on the interpretation of consciousness in this article.

To sum up, in this paper, we do not propose a self-programming system to address consciousness, but rather we explain consciousness as one of the justifications underpinning the self-programming system as well as the basis for further solutions to other cognitive tasks.

Next, we will first introduce the research background of self-improving AI systems and consciousness. Meanwhile, we will also briefly demonstrate how the self-programming system can solve these two problems. Then, in the second chapter, we will detail the runtime framework of the self-programming system, including its basic setting, runtime environment, and learning mechanism.

1.1 Self-improving system

Since artificial intelligence was proposed, self-improvement has been regarded as one of the core goals that a true AI must meet. Turing envisioned that instead of directly

writing countless functional programs, it would be better to write a program that could imitate a child and then train it to the level of intelligence as an adult. (Turing, 1950) This process is same as educating a child to an adult. Later, another founder of AI, John McCarthy, summed up systems with the ability described by Turing as self-improving systems. And, he further believes that self-improvement is a key capability of true artificial intelligence. (Hall, 2007; McCathy, Minsky, Rochester, & Shannon, 1955)

In the early stages of artificial intelligence, there is no shortage of researchers pursuing AI following Turing's intuition. Commonly, their projects start from a basic cognitive structure and then build knowledge by learning. This learned knowledge further helps to solve more problems and accumulate new knowledge, such as GPS, AM and so on. However, these attempts were unsuccessful. (Lenat & Brown, 1984; Lenat, 1982) All of them ended up with a similar dilemma, that is the farther away from the presetting knowledge, the less efficient the searching for new knowledge. Taking the AM system as an example (Automatic Mathematician, a system that simulates mathematicians automatically discovering new mathematical concepts), 125 of the 200 earliest established mathematical concepts are considered meaningful, and the hit rate is 62.5%. However, only 10% of the next 300 concepts were considered meaningful. In other words, the common problem faced by early AI systems is that the returns of their learning are diminishing.

In this case, researchers quickly realized that to solve this problem, learning must not only accumulate knowledge but also must make learning methods better. Following this direction, they have made some attempts, such as the Eurisko, an improved version of AM. The key improvement of Eurisko is that it tried to allow the system to summarize past experiences of discovering new rules and thus form high-order rules for discovering new rules (Heuristics). If this ability can be realized, then Eurisko is no longer an AI system of a special field, but a general AI system. However, this attempt failed attributed to the abysmal distance from achieving the human-level learning ability which is expected by the designers.

In addition to Eurisko, there are also other models that try to achieve self-improvement by other means, for example, the Godel Machine proposed by Jurgen

Schimidhuber (2003) and the AIXI model of Marcus Hutter (2000). A common feature of both these models is that they improve behavior by maximizing the expectation of agent's utility. To these models, the generally accepted dilemma is that although they are theoretically achievable, there are enormous difficulties in actually implementing them.

However, in our viewpoint, all the above models are facing the obstacle much more than just a matter of implementation or performance. They have fatal flaws in their initial design.

The problem with the Eurisko-like models is that it is not enough to just summarize the rules for discovering rules. If there are rules for discovering rules, do we also need to discover rules for discovering rules for discovering rules, namely the second-order rules, third order rules and so on? In this sense, such a process is endless. This endless cycle of self-referentiality is a key feature of intelligence and even consciousness, as Hofstadter (1979; 2007) concludes.

Faced with this critical issue, Eurisko has circumvented it by presupposing more abstract axioms. For example, Eurisko adopted the following two axioms as part of its basic settings. 1) A rule is time-invariant. 2) The scopes of rules are based on similarity. But these presuppositions are inherently unreliable. To the first axiom, rules in practical situations frequently change over time. Even a relatively immutable rule like a legal clause cannot be said to be immutable forever. To the second one, the assumption that scope depends on similarity is also problematic, since similarity itself is just a result of an intelligent phenomenon that needs to be explained.

The problem with all models that rely on utility optimization is that humans don't know what the utility of a process or outcome will be until they try it out. The premise of building a complete utility formula is that the system must have tried all the processes and outcomes that have positive and negative utility. This has been ignored by the model's builders who exogenous assume that all these utilities are known and independent of each other.

We believe that the key to building a self-improving AI is not to come up with some specific algorithm, but to rely on a suitable storage system as we propose in our

previous article. Depending on this storage system, the behavior and phenomena in any specific domain are just some information that has been stored in it. In other words, the appropriate response to a special environment can be abstracted by comparison with the storage system. This process can abstract the specific environment and behavior to the properties that their corresponding storage objects satisfy in the storage system. In this case, the endless self-referential loop is no longer a problem, because as long as it is stored in a suitable way, the rules abstracted at different levels can be applied synthetically.

There is another benefit of using the rules abstracted by the storage system. It can turn the operation of the whole system into a sequence of certain operations under certain conditions. This process is the same as the conditioned reflex in biology, so it is also the most biologically plausible way. In the meantime, it avoids the presetting of the basic structure of a world and its utility function like a system that relies on a maximizing utility, thereby also avoiding the computational problem of maximizing utility functions.

1.2 Consciousness

How does consciousness exist? This question, like how intelligence works, has haunted all intellectuals since ancient history. However, due to its difficulty, it had been excluded from the field of science for a long time. Even worse, it was excluded from discussion by advocates of positivism, along with metaphysics. Until recent decades, attributed to the accumulation of empirical conclusions about how the brain works from neuroscience, scholars began to try to reason consciousness and proposed various theories (Seth & Bayne, 2002). Among these theories, one major category is constructed from the perspective of brain function. For example, the Global Workspace Theory (GWT) regards consciousness as a global space for information interaction. (Baars, 1988, 1997, 2002; Dehaene & Changeux, 2011; Mashour, Roelfsema, Changeux & Dehaene, 2020) The information in it will be broadcast to various subsystems, thus these subsystems can be combined to determine the optimal behavior globally.

Another class of influential theory from the functional perspective is the higher-order theory (HOT). The core idea of these theories is that if some information is conscious, then it must be the information for meta-representation. (Brown, Lau, & LeDoux, 2019; Rosenthal, 2005) The meta-representation here refers to a description that is not a direct description of the world but a higher-level description that goes beyond objective facts. For example, "yesterday, the vase was broken and seriously affected my mood." In this case, the broken vase is a description of the objective world, and the whole sentence is a meta-representation beyond the objective.

In addition to these theories that view consciousness as a function of the brain, there is also a class of theories that argues that consciousness arises from some structures, and as long as these structures exist, it is conscious. For example, the well-known Integrated Information Theory (IIT) falls into this category. (Tononi, 2004; Tononi & Koch, 2015; Tononi et al., 2016) It claims that consciousness research should start from the phenomena of consciousness, and then infer the corresponding structures from these phenomena. Then it concluded that any system with these specific structures has consciousness.

However, if we rely on a framework than can achieve human-level intelligence, there is another way to think about the source of consciousness--why consciousness is indispensable in this framework. Even, we can argue that explaining why consciousness is needed ought to be one of the essential proof for any framework claimed can achieve human-like intelligence.

Specifically, the reason that an AI framework needs consciousness should satisfy the following criteria: First, the existence of consciousness is because it provides some internal meaningful functions. And these functions are essential to this framework. This means that once these functions are missing, the framework cannot work from its beginning, and thus subsequent functions cannot be completed. As a counterexample, a model of a very simple organism that needs only to respond to external conditions has no need for consciousness.

Second, this function is exclusive, namely, the system is inoperable if it is treated in the same way as the unconscious function.

Third, the information expressed by its function is extractable, and this extractability is inherent meaningful for the framework.

In the framework of self-programming systems, consciousness is part of the runtime state space. The information in it will be compared with the information in the storage system. This will extract the abstract relationships based on the storage system. Then these abstract relationships will spontaneously trigger operations that can manipulate the storage system such as retrieval of corresponding information.

In the self-programming system, abstracting based on the storage system is the core of realizing self-improvement. And also all learning in this system depends on such abstraction. So its indispensability is obvious.

Second, because the process of abstraction is automatic and repeatable, If there is no distinction between the unconscious state and the conscious state, the abstract process will repeat infinitely. In other words, it will fall into an endless loop. Consciousness is therefore exclusive.

Third, since the operation of the self-programming system is similar to the invocation of functions in computer programming, the new operation often needs to rely on the past operation results. Therefore, the extraction of conscious information from the past is inherent significant to the system.

2. The Runtime Framework of the Self-programming System

In our previous article, we introduced a storage system. In this article, we will further introduce how to use this storage system to implement the runtime framework of the self-programming system. Specifically, we will divide the following content into three parts: 1) Define the components in this framework. 2) Explain how the self-programming system runs. 3) Introduce its learning mechanism.

2.1 Basic operations and basic elements

The components we first introduce are Basic Operations and Basic Elements. In the general-purpose computer, basic operations and basic elements refer to some basic

symbols that are preset in the computer language. For example, logical operations like NAND or mathematical operations like addition and subtraction. The basic elements generally refer to symbols that can be manipulated such as numbers and identifiers. In other words, these computers are essentially defined on the basis of logic and mathematics. But in our framework, Basic Elements and Basic Operations have completely different meanings from that of traditional computers.

Specifically, both Basic Operations and Basic Elements refer to certain identifiable basic operations or signals provided by peripherals. These peripherals can refer to a certain part of the body, or they can refer to a module in the brain, such as a module that generates emotions.

So what are the Basic Operations and Basic Elements that peripherals provide? Generally speaking, since the functions of each peripheral are different, the basic operations and basic elements provided by each peripheral are also different. For the eyes, the basic operations can be rotation, positioning, focusing, and so on. The basic elements of the eye can be certain color blocks or a specific shape. For limbs, the basic operation can be some kind of rotation or movement. The basic elements can be moving to a certain angle or some tactile signal and so on.

In the self-programming system, although basic operations and basic elements are the basis for the thinking process, they are not so critical. This non-criticality is reflected in the fact that the system can display the same level of intelligence even without certain peripherals. There is only one category of basic operations that is indispensable here, which is the operations provided by the storage system.

Specifically, it only needs to have a storage system and any one way to interact with the external world, then it can produce an effective representation. Next, we will formally define components that are related to building these representations.

The above view of basic elements and operations is somewhat similar to embodied cognition in philosophy and psychology. However, there is a key difference from the current mainstream embodied cognition.

Current mainstream scholars believe that embodied cognition and symbolic manipulation contradict with each other (Varela, Rosch, Thompson, 1991; Shapiro and

Spaulding, 2021). We think this belief is due to the lack of self-programming. The symbol-manipulation-based self-programming system can be used to replace the connections algorithm in embodied cognition. *This substitution will be not only a new method of realizing embodied cognition but also enables the mind and body to operate under the same frameworks so that one of the most important problems in philosophy, the mind-body problem, can also be solved.*

2.2 Operations, Properties, Property set and Storage system

We first give the definitions of the following four concepts, and then make further analysis on this basis:

Operations: a sequence of other operations or basic operations that can be executed under specific conditions; these specific conditions here refer to the object that can be operated must have certain properties.

An object has a certain property: if places a certain operation on this object, it will inevitably obtain another object that definitely possesses all properties of a certain property set or have a property represented by a basic element.

Property set: A collection of one or more specific properties. It is the basic storage object in the storage system.

Storage system: It consists of two parts, one is a collection of all property sets, and the other is some specific operations that can retrieve and compare information stored in this storage system.

At first glance, the above definition seems to have a circular definition problem. For example, the definition of an operation depends on a condition, and a condition is a property, but on the other hand, the definition of a property depends on the definition of operation. In addition, the definition of a property set depends on the property, and the definition of the property itself depends on other property sets. However, if we think in terms of construction, the above definition is logically clear.

The reason is that these definitions can be built up step by step starting from basic elements and basic operations. Specifically, the combination of basic elements and basic operations is sufficient to construct a sequence of operations and their results.

Thereby, properties are constructed. And multiple properties actually form a set of conditions, which can be combined with a sequence of other basic operations to form a new operation. In other words, the conditions of an operation are actually constructed gradually in order, that is, the properties constructed first become the conditions under which the new operation can be created. The same method can also be used to construct property sets, that is, starting from the property set represented by a single property, and gradually defining more complex property sets. (See Figure 1)

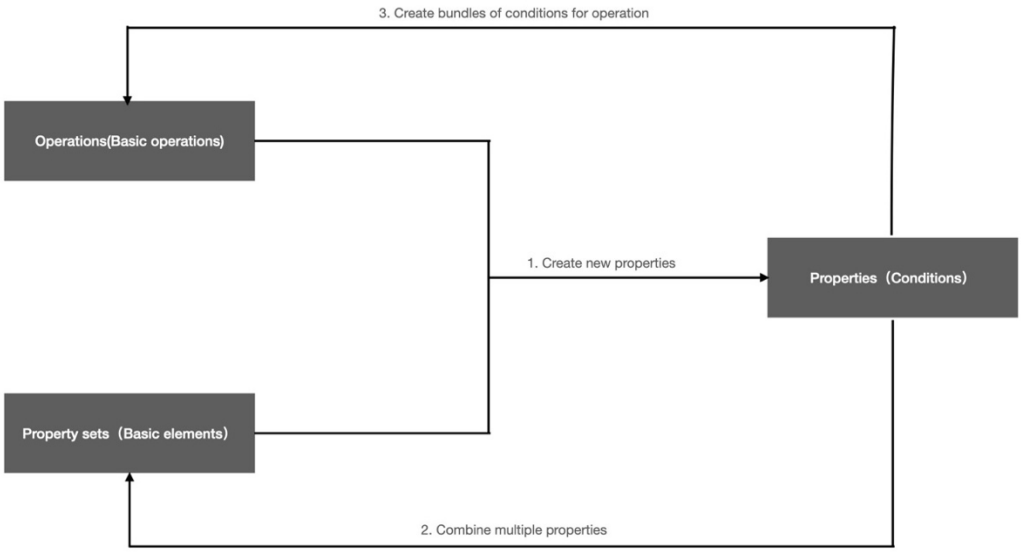

Figure 1 The relationship between operations, properties and property set

2.3 The runtime of the self-programming system

Next, let's take a look at how the self-programming system utilizes its storage system to run. The running of a self-programming system can be summed up in one sentence: it is a mapping from a runtime state to an operation. We have already talked about the definition of operation, but what is the runtime state?

First, at any given moment, the runtime states can be divided into two parts, the explicit state and the implicit state. An explicit state refers to a set of states that express the external world and internal states that are currently perceived through observation, perception, or computation. For example, if someone saw a plate on the table with an apple in it, his/her explicit state will include these property sets that represent the apple, the plate, and the table, and the network that represents the positional relationship

between these three. In this way, the explicit state represented the observed state of the external world. The internal state express, for example, the current mood or the feeling like hunger in the body. At the same time, in the explicit state, there is also a goal. For example, when you are hungry, the goal can be to find a way to eliminate hunger. In a similar way, the explicit state can also represent relationships that have a time span, such as, the "self" just tried to solve a problem in a certain way but failed.

Then what is the implicit state? Simply speaking, the implicit state is the relationship between the explicit state and the storage objects in the current storage system. For example, let's say the current explicit state is that there is an apple on the table as described above, and the goal is to eliminate hunger. Then the implicit state may be: all storage objects that represent apples in the storage system can eliminate hunger by "eating it" (state 1); it could also be: there are some storage objects that represent apples indicate that apples can eliminate hunger, but others indicated not, such as existing a storage object representing a toy apple. (state 2).

On the surface, there seems no essential difference between implicit and explicit state, since both of them are established through some kind of mental operation. But in fact, there are two differences between them. First, the transition from explicit state to implicit state is spontaneous. Without distinguishing between implicit and explicit, the runtime state will grow indefinitely. This is because the current explicit state generates the implicit state, the implicit state can continue to be compared with the objects in the storage system, resulting in a second-order implicit state. If there is no controlling, this process will continue endlessly. Once these two states are separated, the process of generating implicit states runs only when the information is deliberately put into the explicit state. In other words, this process can be called if and only if it is necessary.

On the other hand, the process of generating an implicit state from an explicit state is parallel and imperceptible. It is like the inherent function of the storage system. This means that the process and the outcoming implicit states are not recorded inside the self-programming system. Thus, there has no way to recall what the implicit state once existed.

Based on the relationship between explicit state and implicit state, we suggest that explicit states are the space of consciousness. Also, there should be a peripheral in the brain that automatically records the history of explicit states. The history of these explicit states can be traced back by using some specific operations. Therefore, we can and can only be aware of the information that once appeared in the explicit state, but cannot perceive the information in the implicit state.

After discussing the runtime states, we can go back and understand the runtime itself. As we said earlier the runtime itself is an ongoing process of mapping from the current runtime state to a specific operation. Now we can discuss what this mapping exactly is.

As we discussed earlier, at a given moment, there are runtime states which contain explicit and implicit state. Then, how does the self-programming system use these runtime states? From a perspective of the runtime procedure, the runtime states will be first mapped to an implicit operation. The role of this implicit operation is to find the appropriate operations that were recorded in the storage system, namely explicit operation, for the current runtime state. And also the implicit operation will determine how to use these explicit operations, such as direct execution or sending to the explicit state, etc. Since explicit operations are recorded by the storage object, an implicit operation that extracts an explicit operation is an operation that acts on the storage system itself. (see Figure 2)

For example, if the implicit operation corresponding to the implicit state happens to find that there is only one explicit operation that can achieve the goal in the explicit state (as in the case of state 1 in the previous example). Then the implicit operation can choose to run this explicit operation directly.

What if the implicit operation find not a single appropriate explicit operation? In some situations, there may exist multiple ways to achieve the goal? For example, if you want to calculate 324x99, you can directly use the general multiplication method, but you can also use 324x100-324 to calculate; Similarly, there may not exist any known operations in the storage system that can achieve the goal, for example, the goals like

how a light-speed spacecraft can be built. There may also exist some way that can only achieve the goal with uncertainty, such as state 2 in the previous example.

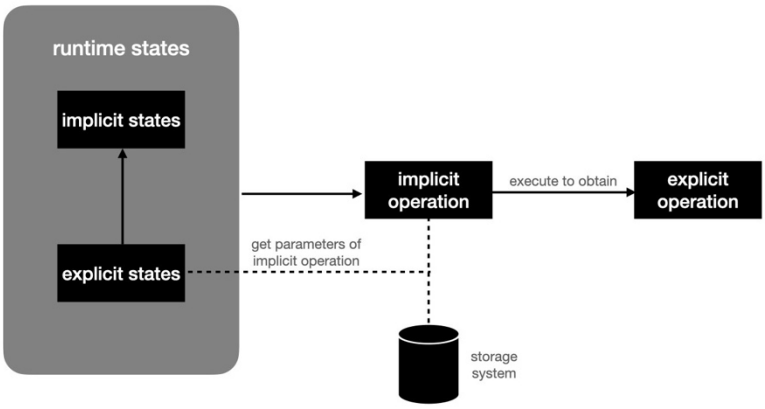

Figure 2 The procedure of runtime

In each of the above situations, there are further subdivisions. For example, in the case of State 2 mentioned above, the implicit operation may choose the explicit operation based on whether there are properties that can be easily collected and helpful for making further decisions. If such a property exists it can execute the explicit operation that can collect this property at first. Corresponding to State 2 of the previous case, it is possible to touch the apple first and decide whether to eat it.

In some cases, the state of the explicit operations discovered by the implicit operation can also be put into the explicit state for further calculations of what should be done. For example, if no possible solution is found, some attempts may be made by using the functions provided by other peripherals, such as a search that allows combining two operations together.

In cases where there are multiple explicit operations, it is also possible to put all these explicit operations into an explicit state to determine which one is more appropriate.

We can see that the process of putting the result found by the implicit operation corresponding to the implicit state into the explicit state is a process that can go on in an infinite loop. So is it possible to get stuck in an endless loop of thinking without actually solving any problem? In theory, this possibility does exist. But in fact, the self-programming system will try to avoid this problem in some ways.

For example, if the same content is written to an explicit state multiple times, a state will be generated in the implicit state, that is the multiple writes in the explicit state are the same. Then, relying on this new state, system can jump out of the infinite loop by give up. For example, reasoning the problem of chicken and eggs which comes first can be followed into this category.

So can this way guarantee that the infinite loop will not happen? The answer is no, for example: in some extreme cases, the system may not be able to find that it has entered an infinite loop, or even if it finds that it has entered an infinite loop, its implicit operation chooses to not change, then the system may enter an infinite loop.

However, we believe that this kind of infinite loop is not a design error of the self-programming system, because such an infinite loop also occurs in the thinking process of human. For example, a person who has symptoms of anxiety or depression is dropped into such a situation. Taking anxiety disorders as an example, it is that the anxiety state will guide the implicit operation to find the anxiety source and try to solve it, but when the anxiety source cannot be solved, it will further cause the anxiety state. Thus entering an infinite loop.

In summary, the runtime of a self-programming system provides a function that maps to the execution of specific operations based on conditions and goals. This function is obtained by comparing the current runtime state with the information in the storage system. Therefore, the whole process of locating and executing a specific operation from the runtime state can be regarded as a basic operation (BO) provided by the storage system. Since an operation in a storage system is a composition of basic operations, this means that the operation that invokes the runtime can actually also be a possible component of the operation that compose properties.

If we compare this point to computer programming, the storage system is equivalent to providing a dynamic mapping from function names to function implementations. This mapping will be automatically updated as the storage objects in the storage system are updated.

More importantly, we can see that when an operation of call runtime procedure is a component of another operation of call runtime procedure, the calling of the parent operation will cause the child operation to be called. Combining this mechanism with the nature of causality introduced in our previous article, and seeing the self as a storage object, then a motivation-driven chain of causality can be formed, that is, the reason that executes the child operations is because of the call of the parent operation. For example, why someone is booking a flight might be because he wants to travel to relax. In this case, booking a flight is a child operation determined by the operation of travel.

Combining the content of our previous article, we can see that no matter the causality of the objective world or the causality of human behavior, all of them are consistent with our theory of causality, that is, the causality is a description of the modification of default operation.

Through the study of the self-programming system, we can discover some important properties. First, a self-programming system is by no means a combination of multiple domain-specified systems. The reason is that the key to realizing a self-programming system is the relationship between the storage system and external observations, and how to operate the data in the storage system under these relationships. This is a completely abstract domain that is independent of any specific domain. No matter what domain a problem belongs to, it ultimately lies in how to manipulate the data in the storage system. This means that, for any information, as long as it can be stored, it can be processed in the same way.

On the other hand, we can see that when the runtime state triggers an operation, the operation could consist of a sequence of sub-operations that may trigger new mappings. This is a process similar to fractal problems in complex science. Therefore, solving one part of a problem is no easier than the whole problem. In other words, without a proper

understanding of the storage system, even trying to solve some seemingly simple problems will lead to clueless.

Third, sample-based methods like current machine learning cannot learn the essence of the self-programming systems. Because the sample-based method can only discover specialized mapping of implicit operations. These mappings are just products of storage-system-related general implicit mappings. (We will discuss how this could be done in the part of the learning mechanism). The failure of learning the general mappings will lead to machine learning exhibiting poor flexibility in a transient environment.

2.4 Learning mechanism

As can be seen from the previous analysis of runtime, if the mapping of runtime states to implicit operations and the information in the storage system are given, the run of the self-programming system will be determined. In other words, how the self-programming system works depends on the information in the storage system and the implicit mapping. There is a naturally following question that is how the storage objects and implicit mapping are established? Or what is the learning mechanism behind them?

The problem is both simple and complex. First of all, we believe that the mechanism of establishing storage objects and implicit mapping is no different from the establishment of conditioned reflex in biology. Simply put, a new property can be formed if conditions, operations, and result objects are triggered repeatedly. Since the properties are the content of the storage object, creating properties is equivalent to creating new storage objects.

However, an answer like this can only be regarded as a basic functional explanation of the learning mechanism. The more important question is, what are the application conditions of the above mechanism or when will this mechanism be used? Unfortunately, facing this question, we can only answer part of it. The other part cannot be summed up by the nature of the self-programming system.

In the self-programming system, the application of any function has two different levels, namely the spontaneous level and the purposeful level. This rule is also

applicable to the learning mechanism. Its spontaneous level refers to the fact that this learning mechanism is automatically triggered during the operation of the system. The role of the learning mechanism at this spontaneous level is relatively simple and can be described. It works on at least the following three aspects.

First, the most immediate aspect is to work with explicit state at runtime. Specifically, if a certain storage object happens to be triggered at some point, its properties are loaded into the explicit state. At this time, if the same result that generated by an operation happened repeatedly, then a new property that contains the new operation and the result will be created. And this new property combines with the properties from the original object to generate a new storage object.

Second, since the runtime state not only has explicit state and explicit operations, but also has corresponding implicit states and implicit operations, the learning mechanism works should also work on the implicit aspect. In the implicit aspect, learning refers to building mappings from implicit states to appropriate implicit operations. Taking the previous calculation 324x99= as an example, the implicit state is that there are multiple ways to calculate this result, and the implicit operation is to list this method into the explicit state and consider it further.

Third, specializing implicit mappings. We introduce this aspect by an example. Assume there is a problem, and both operations A and B known in the system can solve it. We know that in this case both operations A and B shall be put into the explicit states to be evaluated by a more general implicit operation. Here, we further assume that the result of the evaluation is that Operation A executes faster so Operation A is always called in more urgent situations; Operation B has a higher success rate, thus it is always called in situations with spare time. Then if these operations are called repeatedly, two new implicit mappings will be created: Calls Operation A under emergency situation. Call Operation B when there is spare time. In this way, the process of loading the implicit state into the explicit state is avoided by forming a specialized mapping, thereby reducing the computational cost.

After talking about spontaneous learning, let's talk about purposely learning. As we said before, if certain states, operations, and results occur repeatedly, then a new

storage object will be generated. This newly created storage object expresses a specific function by its properties. The learning mechanism can still be viewed as a function, thus it can also be expressed by a storage object which is created by the repeat of the spontaneous learning process. The result is that a storage object that expresses the learning mechanism will exist in the storage system.

Once the above storage object is created, the self-programming system can use the learning mechanism to create new storage objects purposefully like other peripherals. In this case of purposely learning, the question of when to apply the learning mechanism becomes a non-summerizable question, since its application conditions are completely determined by the self-programming system itself. As we said earlier, the problem of self-programming is a fractal problem. So in this sense, summarizing it is equivalent to resummarizing the whole self-programming system.

2.5 Why does self-improvement can be achieved?

We have discussed earlier that the core problem with methods like Eurisko which attempt to achieve self-improvement by automatic summarizing heuristics is that the performance of the newly created heuristics gradually decreases. The essence behind this problem is actually the problem of Hume's induction, namely that past validity does not guarantee its future validity. We have discussed this in detail in our previous article.

Let's imagine if we apply this point to a system like Eurisko, what will happen? Obviously, heuristics generated will become more and more ineffective and decay to no effect at all. This is consistent with the empirical findings.

A common misconception here is that as long as an assumption is weak enough, it can be considered irrelevant. But we don't think so. A generally weak hypothesis, still much stronger than a complete unknown. For example, it is often considered a weak assumption that the events in the world follow some completely unknown distribution. But in fact, this assumption at least implied that the distribution is time-invariant. But this point cannot be justified at all.

More generally, all attempts to formally justify the rationality of an intelligent model are impossible when the agent makes no prior assumptions about the external

world. Because any formal justification is essentially a deduction from an agent's behavior to some plausible properties that were defined in a model that represents the external world. This means that the verification of human-like intelligence should be somewhat subjective, such as the Turing test. However, the problem with this kind of subjective verification is that it can only be applied to scenarios where humans basically agree with each other, for example, face recognition. However, it is not suitable for verifying the theory of the mind's high-level abilities, such as judgment, reasoning, decision-making, etc.

Based on this situation, we believe that there is a third way to verify the theory of human-level AI. This approach is to exploits the generality of human-level AI. Specifically, it is to compare the theory of human-level AI with theories describing a certain aspect in various domains. If this theory of human-level AI can combine and complement existing theories in these domains, then it should have plausibility to some degree.

Now, let’s go back to our comparison with traditional models. The principle of the self-programming system is completely different from theirs. Under this system, the core goal of learning is not to try to find the rule of the external world, but how the storage system works. This means the object that needs to be discovered changes from an open one that can never be fully observed and can change over time to a time-invariant closed system.

In this case, the mechanical stability of the storage system guarantees the validity of the relationship between implicit conditions and implicit operations. For example, from the swans seen in the past that are all white, we cannot infer that all swans in the world are white. But, under the current known information, the validity of the corresponding operations to this abstract state, that is --when there is no counter-example in the storage system, the system should search more information in the storage system to support decision, does not change. This internal stable validity guarantees the validity of the method of exploring the world. And the stable validity of exploring the world is actually what is called self-improvement.

2.6 Comparing with other theories of consciousness

As pointed out at the beginning, we have found a necessary reason for the existence of consciousness from the perspective of creating the mind, so as to find a computational explanation of consciousness within the self-programming system. If this explanation is correct, it should be able to integrate with existing theories of consciousness, complement what is missing, and help analyze where they are complementary and where they are conflicting with each other. Next, we will look at the relationship between our theory of consciousness and existing theories of consciousness.

First and foremost, we believe that the three types of consciousness theories described in the introduction do not have any essential conflict in their starting points. In our interpretation, the role of the conscious space is to compare with the storage system for abstracting the relationship in the storage system. This abstracted information will further trigger operations on the storage system. Consciousness, therefore, is both a product of structure and at the same time intended to achieve some kind of meta-representation. This is consistent with both these theories that hold that consciousness is based on a structure, as well as with higher-order theories.

Moreover, because the storage system stores functional information aggregated from the brain、body and even other tools, it is able to determine appropriate behavior globally. From this point of view, it is consistent with the Global Workspace Theory. So, from our point of view, the starting points of these three theories just describe three different perspectives of consciousness.

Second, we can also make the understanding of higher-order theory and global space theory clearer. Specifically, what is the meta-representation in the HOTs (Brown, 2015; Cleeremans, 2011; Cleeremans et al. 2020; Fleming, 2020; Lau & Rosenthal, 2011; LeDoux & Brown, 2017) and what does the "global" in the GWT refer to? (Bayne, 2010; Carruthers, 2019) These two problems have always been one of the core problems they face respectively.

In our theory of consciousness, both these points can be clearly explained. The core of the meta-representation is the relationship of the conscious content to the storage

system. The global space in the global space theory is actually the space used to extract abstract relationships based on the storage system.

Third, in addition to making existing theories more explicit, our theory can also provide phenomenal explanations for functional theories like GWT and HOTs. Specifically, some information that can be conscious actually means that it is in the explicit state space and can be retrieved. Why does this information in the explicit space need to be kept and retrievable? This is because the information in the explicit state space needs to support implicit operations and subsequent explicit operations. So they need to be maintained over a longer period of time. Thus this information can also be retrieved by operations that can examine the explicit states.

Finally, our theory can also directly explain the unity of consciousness (Bayne, 2010; Bayne & Chamlers, 2003)—that is, why the information that is realized is always integrated rather than separated aspects. In our theory, the role of information in consciousness is to correspond to the storage objects in the storage system, and each object is a complex of multiple properties. Therefore, what consciousness must perceive is not the individual properties, but the integration of them.

3. Summary and Outlook

Combining our last article's discussion of induction/deduction, causality and the discussion of self-improvement and consciousness in this article, we have shown the power of the self-programming system theory on four domains, thus indirectly demonstrating its rationality. While this approach may not be as convincing as direct verification of the new insights our theory offers, we believe that this cross-domain agreement with existing theories and evidence should never be dismissed as trivial coincide.

Based on the storage system established in the previous article, in this article, we introduce the runtime environment and learning mechanism of the self-programming system. One of our key arguments is that various functions of the body and brain are recorded in storage systems in abstract form. In this way, when a problem needs to be

solved, it can be planned through the storage system. These recordings can also be extended to tools outside the body. In this sense, we collectively call these tools that can be recorded by the storage system as peripherals. For example, the learning mechanism described in this article is a key peripheral in the brain. Our claim is actually consistent with the extended cognition's viewpoint (Clark and Chamlers, 1998; Menary 2010). And more than that, the self-programming system provides the logic behind the phenomena of extended cognition. Specifically, such phenomena are consequences of the self-programming system's interaction with the outside world.

So what other peripherals do our brains and bodies have? Some peripherals are very obvious, for example, eyes, ears, nose, legs, and other parts of our bodies. But there are also peripherals that are hidden behind the scenes like learning mechanisms, such as emotions and recall, etc. The analysis of what these peripherals are and how they work will be gradually expanded in our subsequent articles.

Reference

1. Baars, B. J. (1988). *A Cognitive Theory of Consciousness*. Cambridge University Press.

2. Baars, B. J. (1997). In the theatre of consciousness: Global workspace theory, a rigorous scientific theory of consciousness. *Journal of Consciousness Studies, 4*(4), 292–309.

3. Baars, B. J. (2002). The conscious access hypothesis: Origins and recent evidence. *Trends in Cognitive Sciences, 6*(1), 47–52. https://doi.org/10.1016/S1364-6613(00)01819-2

4. Bayne, T. J. (2010). *The Unity of Consciousness.* Oxford University Press UK.

5. Bayne, T. J. & Chalmers, D. J. (2003). What is the unity of consciousness? In Axel Cleeremans (ed.), *The Unity of Consciousness*. Oxford University Press.

6. Brown, R. (2015). The HOROR theory of phenomenal consciousness. *Philos Stud* **172,** 1783–1794. https://doi.org/10.1007/s11098-014-0388-7

7. Brown, R., Lau, H., & LeDoux, J. E. (2019). Understanding the higher-order approach to consciousness. *Trends in Cognitive Sciences, 23*(9), 754–768. https://doi.org/10.1016/j.tics.2019.06.009

8. Carruthers, P. (2019). *Human and animal minds: The consciousness questions laid to rest.* Oxford University Press. https://doi.org/10.1093/oso/9780198843702.001.0001

9. Clark, Andy & Chalmers, David J. (1998). The extended mind. *Analysis* 58 (1):7-19.

10. Menary, R. (Ed.). (2010). *The extended mind.* MIT Press. https://doi.org/10.7551/mitpress/9780262014038.001.0001

11. Cleeremans A. (2011). The Radical Plasticity Thesis: How the Brain Learns to be Conscious. *Frontiers in psychology*, *2*, 86. https://doi.org/10.3389/fpsyg.2011.00086

12. Cleeremans, A., Achoui, D., Beauny, A., Keuninckx, L., Martin, J. R., Muñoz-Moldes, S., Vuillaume, L., & de Heering, A. (2020). Learning to Be Conscious. *Trends in cognitive sciences*, *24*(2), 112–123. https://doi.org/10.1016/j.tics.2019.11.011

13. Dehaene, S., & Changeux, J. P. (2011). Experimental and theoretical approaches to conscious processing. *Neuron*, *70*(2), 200–227. https://doi.org/10.1016/j.neuron.2011.03.018

14. Dreyfus, H. L. (1972). *What computers can't do: A critique of artificial reason.* New York: Harper & Row.

15. Fleming S. M. (2020). Awareness as inference in a higher-order state space. *Neuroscience of consciousness*, *2020*(1), niz020. https://doi.org/10.1093/nc/niz020

16. Hall, J.S.(2007). Self-improving AI: an Analysis. *Minds & Machines* **17,** 249–259. https://doi.org/10.1007/s11023-007-9065-3

17. Hofstadter, D. (2007). *I am a strange loop.* Basic Books.

18. Hofstadter, Douglas R., 1945-. (1979). *Gödel, Escher, Bach : an eternal golden braid.* New York :Basic Books,

19. Hutter, M. (2000). A Theory of Universal Artificial Intelligence based on Algorithmic Complexity. *arXiv e-prints*, cs/0004001. Ανακτήθηκε από http://arxiv.org/abs/cs/0004001

20. Lau, H., & Rosenthal, D. (2011). Empirical support for higher-order theories of conscious awareness. *Trends in Cognitive Sciences, 15*(8), 365–373. https://doi.org/10.1016/j.tics.2011.05.009

21. LeDoux, J. E., & Brown, R. (2017). A higher-order theory of emotional consciousness. *PNAS Proceedings of the National Academy of Sciences of the United States of America, 114*(10), e2016–e2025. https://doi.org/10.1073/pnas.1619316114

22. Lenat, D.B. & Brown, John Seely (1984). Why am and eurisko appear to work. *Artificial Intelligence* 23 (3):269-294.

23. Lenat, D.B. (1982). The Nature of Heuristics. *Artif. Intell., 19*, 189-249.

24. Li, F. (2022). Why is the mind a self-programming system?. https://doi.org/10.31234/osf.io/jvs58

25. Mashour, G. A., Roelfsema, P., Changeux, J. P., & Dehaene, S. (2020). Conscious Processing and the Global Neuronal Workspace Hypothesis. *Neuron*, *105*(5), 776–798. https://doi.org/10.1016/j.neuron.2020.01.026

26. McCarthy, J., Minsky, M. L., Rochester, N. & Shannon, C. E. (1955). A PROPOSAL FOR THE DARTMOUTH SUMMER RESEARCH PROJECT ON ARTIFICIAL INTELLIGENCE http://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html .

27. Rosenthal, David M. (2005). *Consciousness and Mind*. Oxford University Press UK.

28. Schmidhuber, J. (2003). *Goedel Machines: Self-Referential Universal Problem Solvers Making Provably Optimal Self-Improvements*. doi:10.48550/ARXIV.CS/0309048

29. Shapiro, Lawrence and Shannon Spaulding, "Embodied Cognition", *The Stanford Encyclopedia of Philosophy* (Winter 2021 Edition), Edward N. Zalta (ed.), URL = <https://plato.stanford.edu/archives/win2021/entries/embodied-cognition/>.

30. Seth, A.K., Bayne, T. (2002) Theories of consciousness. *Nat Rev Neurosci* **23,** 439–452. https://doi.org/10.1038/s41583-022-00587-4

31. Tononi, G. (2004). An information integration theory of consciousness. *BMC Neurosci* **5,** 42 . https://doi.org/10.1186/1471-2202-5-42

32. Tononi, G., & Koch, C. (2015). Consciousness: here, there and everywhere?. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, *370*(1668), 20140167. https://doi.org/10.1098/rstb.2014.0167

33. Tononi, G., Boly, M., Massimini, M. *et al.* Integrated information theory: from consciousness to its physical substrate. *Nat Rev Neurosci* **17,** 450–461 (2016). https://doi.org/10.1038/nrn.2016.44

34. Turing, A. M. (1950). Computing machinery and intelligence. *Mind, 59,* 433–460. https://doi.org/10.1093/mind/LIX.236.433

35. Varela, F. J., Thompson, E., & Rosch, E. (1991). *The embodied mind: Cognitive science and human experience*. Cambridge, Mass: MIT Press.