第 2 届世界人工意识大会热身–媒体与顶刊速递系列

The 2nd World Conference on Artificial Consciousness

第二届世界人工意识大会(DIKWP-AC 2024)

Artificial Consciousness: The Confluence of Intelligence and Consciousness in the Interdisciplinary Domain

The 2nd World Conference on Artificial Consciousness
第二届世界人工意识大会(AC2024)

Artificial Consciousness: The Confluence of Intelligence and Consciousness in the Interdisciplinary Domain

Background

Challenges and Opportunities in the Digital Era

We are currently living in a digitally-driven era with data at its core. Every day, a vast amount of data is continuously generated and processed. Information disseminates globally at unprecedented speeds, knowledge accumulates and evolves, and wisdom manifests its unique value across various domains. However, the challenges posed by this era revolve around ensuring that data processing and transformation are not only technologically efficient but also align with human ethical and moral standards, as well as genuine human purposes. The significance of addressing this issue cannot be understated, as it directly relates to the sustainability of the digital economy and the digitized development of society.

In this digital age, artificial intelligence (AI) technology is rapidly emerging as a primary driving force behind the digital economy. AI systems exhibit unprecedented speed and accuracy in processing vast amounts of data, propelling automation, intelligence, and innovation. However, the development of AI technology is accompanied by a series of significant challenges. Firstly, data rights protection has become a highly scrutinized issue, given its implications for personal privacy and data security. Secondly, the issues of algorithmic discrimination and bias highlight the limitations of technology and the potential threats to social justice. Additionally, the protection of intellectual property rights is crucial for fostering innovation and technological development. These challenges not only impact individuals but also have profound effects on the digital services of businesses and governments, thereby influencing the overall sustainability of society and the economy..

Computer Architecture and Chip Design for DIKWP Artificial Consciousness

Kunguang Wu, Yucong Duan, Liang Chen, Yingbo Li, QiQi

Abstract

Artificial intelligence systems are often accompanied by risks such as uncontrollability and lack of explainability. To mitigate these risks, there is a necessity to develop artificial intelligence systems that are explainable, trustworthy, responsible, and demonstrate consistency in thought and action, which we term Artificial Consciousness (AC) systems. Therefore, grounded in the DIKWP model which integrates fundamental data, information, knowledge, wisdom, and purpose along with the principles of conceptual,

https://doi.org/10.32388/Q5RC1J

cognitive, and semantic spaces, we propose and define the computer architectures, chips, runtime environments, and *DIKWP* language concepts and their implementations under the *DIKWP* framework. Furthermore, in the construction of AC systems, we have surmounted the limitations of traditional programming languages, computer architectures, and hardware-software implementations. The hardware-software integrated platform we propose will facilitate more convenient construction, development, and operation of software systems based on the *DIKWP* theory.

I. Introduction

Despite significant advancements in the fields of computer science and artificial intelligence (AI) research and applications [1], the processing details of existing AI systems remain opaque, potentially introducing unforeseen risks [2]. To construct an explainable, trustworthy, responsible, and consistently acting AI system, termed Artificial Consciousness (AC), we integrate the *DIKWP* (Data, Information, Knowledge, Wisdom, Purpose) concept [5] [6] with the world's first semantic-mathematical definition of consciousness relativity [4] to propose a definition based on AC-related implementations. Furthermore, within the *DIKWP* architecture, we have defined the *DIKWP* language in the *DIKWP* runtime environment and attempted to overcome the programming complexities associated with implementing *DIKWP* transformations in traditional computer architectures, thereby laying the groundwork for building a *DIKWP* hardware-software ecosystem. The *DIKWP* -based AC computer architecture ensures efficiency and controllability. Semantic transformations of *DIKWP* content [7] coupled with AC modeling guarantee consistent verbal and behavioral characteristics within AC systems, thus minimizing semantic loss during human-machine interaction.

Section II primarily reviews the development of software and hardware integration and the necessity of constructing AC systems. It introduces the concept of building an integrated hardware-software platform and AC systems based on the *DIKWP* model, theoretically addressing the current deficiencies in artificial intelligence systems. Section III describes the problem, detailing the complexities of implementing *DIKWP* -related functionalities under the limitations of traditional programming languages, software-hardware integration, and computer architectures. In section IV, we attempt to convert the problems outlined in section three into engineering challenges and propose a series of solutions. Our intent is to resolve these engineering issues theoretically. Section V concludes the paper.

II. Related Works

A. Hardwarelization of software

Researchers have made significant progress in the hardware implementation of software. Ambrogio et al. introduced a chip designed for simulating AI [8], which accelerates computations related to speech recognition while providing excellent performance and lower power consumption. Jesús and colleagues proposed an FPGA-based acceleration circuit [9], which enhances the computation of semantic trees and significantly reduces the execution time of parsing algorithms. Chen et al. introduced a fully analog optoelectronic chip named ACCEL [10], capable of providing high computational speeds and energy efficiency for visual tasks. However, the current manufacturing costs and domain-specificity of these chips are relatively high, limiting the universality of the transition from software to hardware.

B. Computer architecture

In the field of computer architecture, the performance of systems built on the von Neumann architecture is significantly influenced by the speed of information exchange between the Central Processing Unit (CPU) and shared memory [11]. The storage speed within the von Neumann architecture is constrained by factors such as storage architecture, access strategies, and materials [19]. Moreover, the von Neumann architecture was designed for Turing computation, achieving notable speed and precision in numerical processing. However, many phenomena are inherently incomplete, inconsistent, imprecise, and unexpressable, making it challenging to directly use numerical computation to handle these phenomena. Therefore, automating the processing of non-numerical tasks on computers based on the von Neumann architecture is quite difficult. To overcome these challenges, there is a need to improve existing computer architectures or to redesign them entirely.

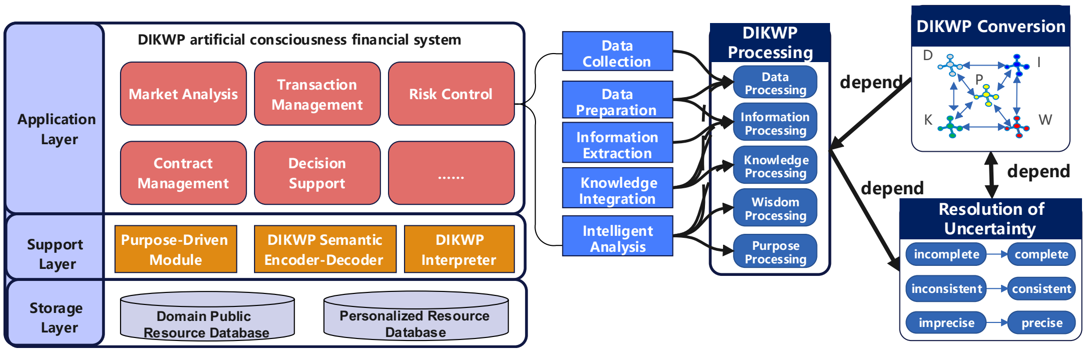
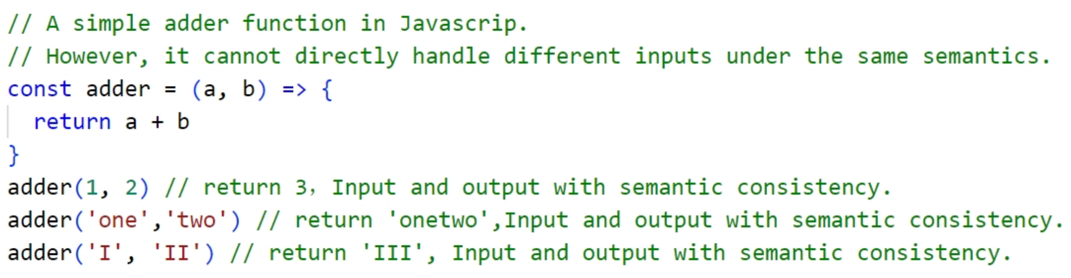


Figure 1- DIKWP-AC Financial System - Processing function mapping (partial)



```
// A simple adder function in Javascrip.
// However, it cannot directly handle different inputs under the same semantics.
const adder = (a, b) => {
  return a + b
}
adder(1, 2) // return 3, Input and output with semantic consistency.
adder('one','two') // return 'onetwo',Input and output with semantic consistency.
adder('I', 'II') // return 'III', Input and output with semantic consistency.
```

Figure 2- A function code of adding ’a’ and ’b’

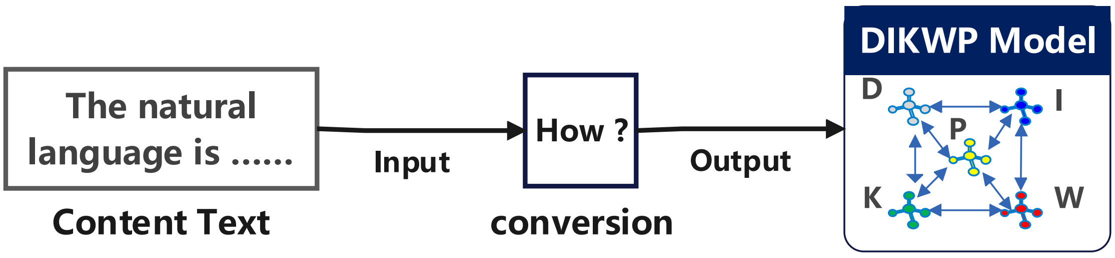


Figure 3- Conversion and mapping of natural language to DIKWP

C. DIKWP-AC

To construct consistent artificial intelligence systems, it is imperative to establish AC systems based on human cognitive modeling and the foundational theories of DIKWP. Human cognitive modeling is a method for studying the cognitive processes of humans, aimed at explaining and predicting human information processing, knowledge acquisition, and decision-making behavior. The DIKWP model offers a novel perspective by emphasizing data, information, knowledge, wisdom, and purpose as core elements for achieving this goal. Duan et al. [13] utilized the DIKWP framework to extend the concept of knowledge graphs to associative data graphs, information graphs, knowledge graphs, wisdom graphs, and intent graphs. To address challenges such as large-scale concept fusion, semantic representation ambiguity, and semantic content confusion, they based their work on the Existence Computing and Reasoning (EXCR) model [14], establishing fundamental semantic relationships under the principle that "Relationships Define Everything of Semantics" (RDXS) through cross- DIKWP modality fusion. Duan et al. used a security protection scenario in edge computing as a research case [15], proposing a formalized semantic perception of key DIKWP elements during the conceptualization process by constructing a meta-model of the DIKWP framework. Gao et al. [16] proposed extending the DIKWP architecture as a resource representation model, providing a systematic approach to constructing entity and relationship elements, thereby enhancing the optimization of service and resource scheduling in multimedia systems. Li et al. [17] expanded upon this work within the DIKWP model, introducing an intent-driven differential privacy architecture within the DIKWP framework, and applied it to domains such as intelligent form filling [18], emotional communication [19], the digital economy [20], biolegal litigation [21], meteorological and depression analysis [22], among others. Concerning DIKWP -AC, Duan presented the working principles of the DIKWP -AC chip [24], compared it with other chips, and summarized the advantages of the DIKWP-AC chip [25]. Lake et al. introduced a new method for studying the generalization capabilities of human systems, demonstrating the potential of neural and symbolic models in achieving human-level generalization capabilities [26]. Duan discussed the role of the DIKWP model in consciousness modeling and brain-machine interface (BMI) processing [28]. Wang et al. [27] proposed a resource invocation processing method for DIKWP-AC systems, pointing the way for the system implementation of AC. Overall, the use of the DIKWP model could potentially overcome the semantic deficiencies between traditional computer architectures, hardware, and programming languages.

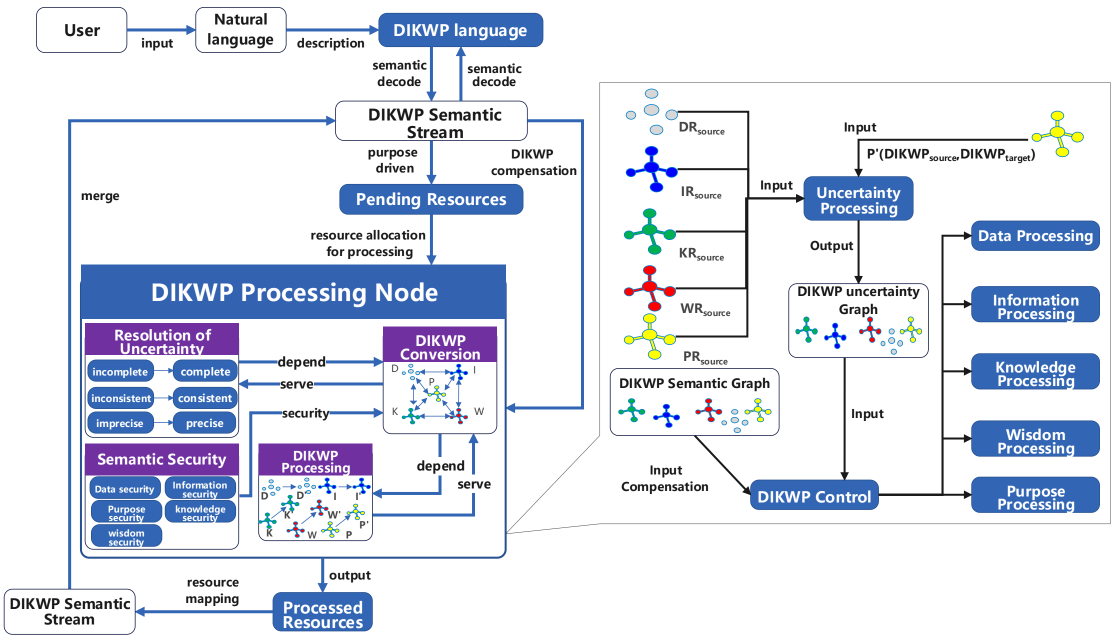


图 4-DIKWP 逻辑架构

D. Concept Space

a) Definition of Concept Space

In Figure 2 on the right, the concept space is depicted as a collection composed of a series of related concepts, interconnected through specific attributes and relationships. Depending on the symmetry of the relationships among the concepts, this space can be represented either as a directed or an undirected graph. Consequently, the concept space can be expressed using the following equation：

$$Graph_{ConC} = (V_{ConC}, E_{ConC}) \qquad 1$$

Where V_{ConC} represents the set of nodes corresponding to concepts, and E_{ConC} is the set of edges denoting the relationships between these concepts.

b) Fundamental Properties

In the concept space, each concept $v \in V_{ConC}$ possesses a set of attributes $A(v)$ and relationships $R(v, v)$ with other concepts.

Attributes: $A(v) = \{a1(v), a2(v), \ldots, an(v)\}$, where each $a_i(v)$ is an attribute of v.

c) Relation

$R(v, v')$represents the relationship between concepts v and v'. If the graph is directed, then $R(v, v')$ is not equivalent to $R(v, v)$; if the graph is undirected, they denote the same relationship.

d) Operation

Operations in the concept space involve a series of actions to query, add, or modify concepts and their relationships within the concept space.

- **Query Operations:** Query operations involve retrieving relevant sets of concepts within the concept space based on a query condition q (such as specific attributes or relationships). The primary expression is as follows:

$$Q(V_{ConC}, E_{Conc}, q) \rightarrow \{v_1, v_2, \dots, v_m\} \quad 2$$

- **Addition Operations:** We can add a new concept v to the concept set V_c using the equation below:

$$Add(V_{ConC}, v) \quad 3$$

- **Modification Operations:** Additionally, we can maintain the relevant attributes of existing concepts through Equation 4.

$$Update\big(V_{ConC}, v, A(v)\big) \quad 4$$

E. Cognitive Space

Cognitive Space (ConN) provides a framework for describing and analyzing cognitive processes, namely how input data or information is transformed into understanding, decision-making, or action. This concept is particularly crucial in handling Data, Information, Knowledge, Wisdom, and Purpose (*DIKWP*), as it reveals how individuals or systems comprehend and respond to the external world through unique cognitive processing. Below is a formalized description of the definition and processing within Cognitive Space.

a) Definition of cognitive space

Functions set: $R = \{f_{ConN_1}, f_{ConN_2}, \cdots, f_{ConN_n}\}$,Herein, each function $f_{ConN_i}: Input_i \rightarrow Output_i$ represents a specific cognitive processing process， where $Input_i$ denotes input space， and $Output_i$ represents the output space.

b) Input Space and Output Space

Input Space $\boldsymbol{Input_i}$ represents the collection of data or information that is perceived, which may include observations from the external world, signals received from other systems, or data generated internally.

Output Space $\boldsymbol{Output_i}$ represents the collection of understanding or decisions formed after processing, which may include the classification of information, the formation of concepts, the determination of intentions, or the setting of action plans.

c) Cognitive Processing Process

Each cognitive processing function f_{ConN_i} can be further refined into a series of sub-steps, including data preprocessing, feature extraction, pattern recognition, logical reasoning, and decision-making, among others. These sub-steps together constitute the complete cognitive pathway from raw data to final output.

Sub-step Representation: For each f_{ConN_i}, it can be represented as:

$$f_{ConN_i} = f_{ConN_{i(5)}} \circ f_{ConN_{i(4)}} \circ \dots \circ fConN_{i(1)} = (Input_i)$$

where $f_{ConN_{i(j)}}$ represents the processing function, j represents j-th sub-step and

$\circ$ denotes the composition of functions.

In the *DIKWP* model, the Cognitive Space transforms data, information, knowledge, wisdom, and intention into concrete understanding and action through the unique cognitive processes of individuals or systems. By invoking various cognitive processing functions, the system can implement the most suitable processing strategies for different types of inputs, achieving efficient and precise decision-making.

F. Semantic Space

Semantic space is a collection composed of a series of semantic units, which are interconnected through specific associations and dependency relationships, collectively forming an objectified representation of information and knowledge. The widely accepted concepts and linguistic rules within semantic space facilitate the transmission and communication of meaning.

a) Definition

We represent this using a graph:

$$Graph_{SemA} = (V_{SemA}, E_{SemA})$$

where V_{SemA} represents semantic units (words, sentences, etc.), and E_{SemA} represents the associations and dependency relationships between semantic units.

b) Semantic Units and Relations

In the semantic space, a series of operations correspond to querying, adding, or modifying semantic units and their relationships.

Query Operation:

$$Query(V_{SemA}, E_{SemA}, q) \rightarrow \{v_1, v_2, \dots, v_m\}$$

The previous equation returns a set of semantic units that satisfy the query condition q.

Add Operation: $Add(V_{SemA}, v)$, adds a new semantic unit v to the set V_{SemA}.

Update Operation: $Update(E_{SemA}, v, v', e)$, updates or adds the relationship e between semantic units v and v'.

c) Operation and Application

Based on the relevant definitions and concepts of the semantic space, we attempt to analyze and address the issues faced by financial firms in executing the legislation regarding HFT mapped into the semantic space as discussed in the previous section. Here, we focus on analyzing the issue of "inconsistency in execution due to content interpretation bias" within the semantic space.

- We define a semantic unit $v_{SEmALawUB}$ to represent interpretation bias, which belongs, to the legal semantic space: $Graph_{SemALaw} = (V_{SemALaw}, E_{SemAlaw})$。
- We can use query operations to retrieve units of inconsistency in the execution process:

$$\boldsymbol{Query}(\boldsymbol{V_{SemA}}, \boldsymbol{E_{SemA}}, \boldsymbol{q_{bias}}) \rightarrow \{v_{SEmALawUB}\} \quad \textbf{5}$$

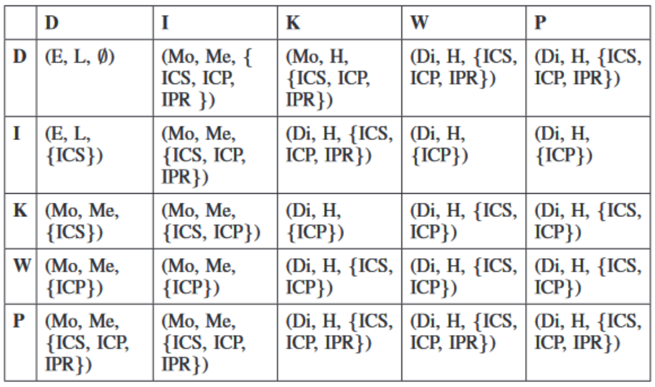
  where condition q is interpretation bias in law.
- The addition operation can be utilized to enrich the semantic space of legal understanding:

$$\boldsymbol{Add}(V_{SemALaw}, v_{SEmALawUC}) \quad \textbf{6}$$

Where $v_{SEmALawUC}$ represents semantic units reflecting accurate legal comprehension.

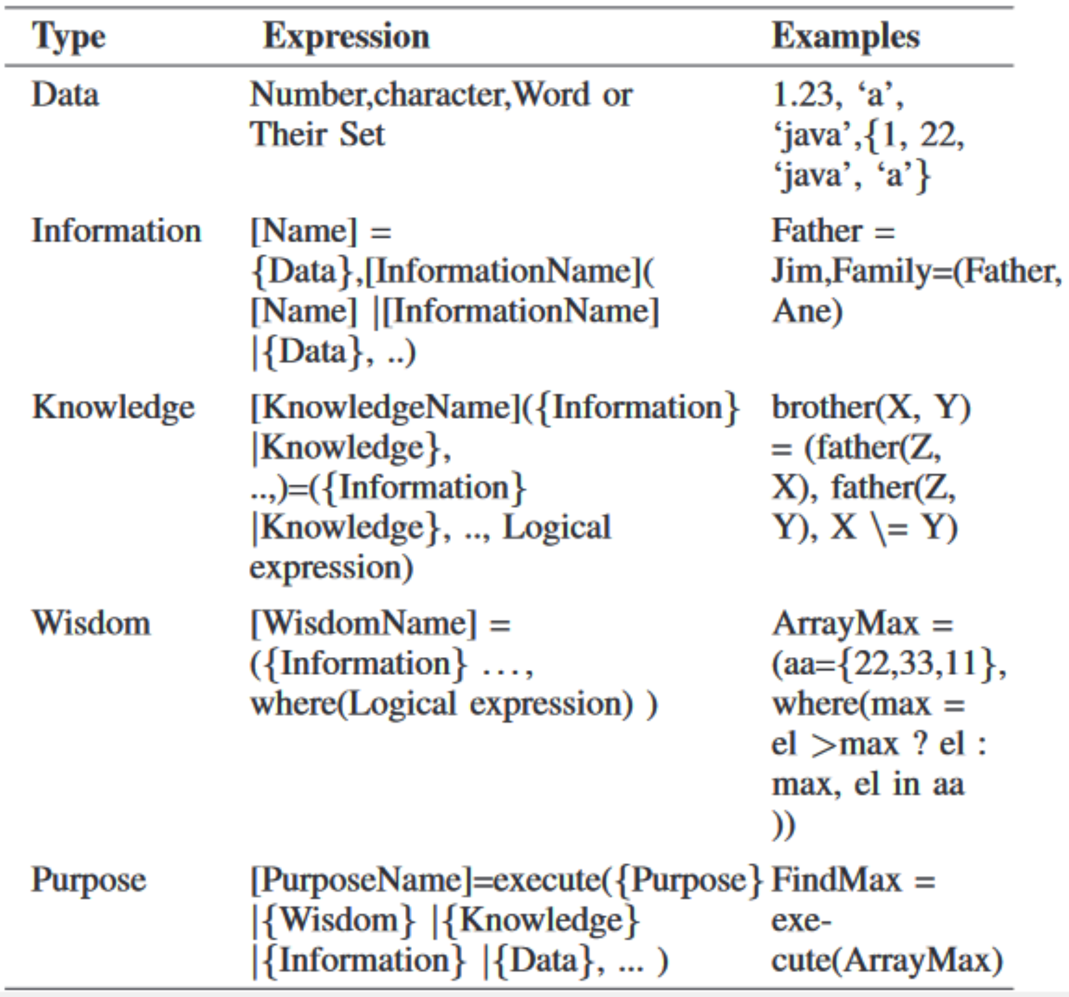
$$\boldsymbol{Update}(V_{SemALaw}, v_{SEmALawUC}, v_{SEmALawUB1}, e_{Ubias}) \quad \textbf{7}$$

- Additionally, the semantic space can be refined through update operations, as demonstrated in Equation 7. The purpose of this operation is to establish new semantic units $v_{SEmALawUC}$ that represent understanding deviations based on the correct legal understanding $v_{SEmALawUB1}$。



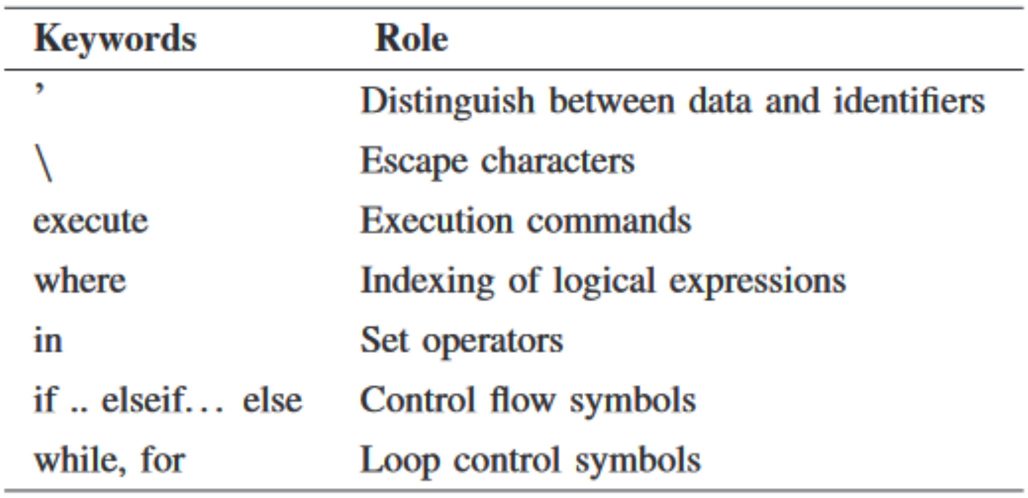
	D	I	K	W	P
D	(E, L, ∅)	(Mo, Me, { ICS, ICP, IPR })	(Mo, H, {ICS, ICP, IPR})	(Di, H, {ICS, ICP, IPR})	(Di, H, {ICS, ICP, IPR})
I	(E, L, {ICS})	(Mo, Me, {ICS, ICP, IPR})	(Di, H, {ICS, ICP, IPR})	(Di, H, {ICP})	(Di, H, {ICP})
K	(Mo, Me, {ICS})	(Mo, Me, {ICS, ICP})	(Di, H, {ICP})	(Di, H, {ICS, ICP})	(Di, H, {ICS, ICP})
W	(Mo, Me, {ICP})	(Mo, Me, {ICP})	(Di, H, {ICS, ICP})	(Di, H, {ICS, ICP})	(Di, H, {ICS, ICP})
P	(Mo, Me, {ICS, ICP, IPR})	(Mo, Me, {ICS, ICP, IPR})	(Di, H, {ICS, ICP, IPR})	(Di, H, {ICS, ICP, IPR})	(Di, H, {ICS, ICP, IPR})

Table 1- Programming difficulty assessment of *DIKWP* conversion



Type	Expression	Examples
Data	Number,character,Word or Their Set	1.23, 'a', 'java',{1, 22, 'java', 'a'}
Information	[Name] = {Data},[InformationName]([Name] |[InformationName] |{Data}, ..)	Father = Jim,Family=(Father, Ane)
Knowledge	[KnowledgeName]({Information} |Knowledge}, ..,)=({Information} |Knowledge}, .., Logical expression)	brother(X, Y) = (father(Z, X), father(Z, Y), X \= Y)
Wisdom	[WisdomName] = ({Information} . . . , where(Logical expression))	ArrayMax = (aa={22,33,11}, where(max = el >max ? el : max, el in aa))
Purpose	[PurposeName]=execute({Purpose} |{Wisdom} |{Knowledge} |{Information} |{Data}, ...)	FindMax = execute(ArrayMax)

Table 2- The expression form of each element in *DIKWP* language



Keywords	Role
’	Distinguish between data and identifiers
\	Escape characters
execute	Execution commands
where	Indexing of logical expressions
in	Set operators
if .. elseif… else	Control flow symbols
while, for	Loop control symbols

Tables 3- Keywords reserved in DIKWP language

III. Problem Description

A. Situation Description

1) **Challenges of Traditional Programming Languages:** Drawing upon the methodologies outlined in [7] for the fusion, and transformation of DIKWP elements under purpose-driven conditions, we endeavor to formalize the conversion and processing procedures into a deterministic program or algorithm. Programming is carried out under the purpose-driven, and the program is executed accordingly. In this study, we assume the existence of a financial system requiring the construction of AC based on DIKWP. As depicted in Figure 1, the Risk Control subsystem comprises five modules, with the execution and processing of their functionalities contingent on transformations between DIKWP elements. From a developer’s perspective, traditional high-level programming languages are employed for programming in this context, with no recourse to machine learning algorithms. In Table I, the y-axis represents the elements of DIKWP input, while the x-axis corresponds to the target type elements for DIKWP conversion. We have defined a triad (PD, CC, IU) to represent the results of an evaluation conducted during the development of AC systems for the programming difficulty (PD), computational complexity (CC), and input uncertainty (IU) assessed as implemented using high-level programming languages (such as JavaScript). The first element of the triad represents the degree of programming difficulty, which includes three evaluation levels: easy (E), moderate (Mo), and difficult (Di). The second element of the triad signifies the computational complexity of the program, categorized as low (L), medium (Me), and high (H). The third element of the triad represents a set of uncertainties in the input DIKWP elements, with elements featuring characteristics of incompleteness (ICP), inconsistency (ICS), and imprecision (IPR). When the inputs simultaneously possess precision, completeness, and consistency, the third element is an empty set.

Table I is evident that the implementation of a financial risk control subsystem is feasible without relying on machine learning. However, both the programming complexity and computational complexity increase as the level of content abstraction rises. Uncertainty is a common issue in the financial domain, necessitating appropriate handling methods. In this process, the determinism of inputs and outputs decreases as the abstraction level increases, and the computational process may require the inclusion of more rules and conditions. Furthermore, we have observed practical difficulties with current programming languages when dealing with uncertain inputs and processing. Specifically, these languages lack the capability to adapt to uncertainty expressed in purpose during runtime. As illustrated in Figure 2, a function for addition is typically confined to inputs in the form of Arabic numerals. If inputs are provided in other languages or different representations, errors may occur. Addressing changes in purpose expression necessitates reprogramming and recompilation for execution. Ensuring interpretability of executable processes at the software level is limited to covering inputs with finite logical steps, which can only accommodate a partial range of semantic expressions. Once inputs are more variations in semantic representations, reprogramming becomes the only viable option. Hence, traditional programming languages face significant challenges in handling uncertainty expressions within the same semantics.

2) **Challenges of hardwarelization of software:** Hardwarelization of software is a design methodology that involves the realization of software functionalities or tasks at the hardware level. In certain specialized domains, this approach, achieved through circuit design, re-implements software functions or tasks on hardware chips to enhance execution efficiency [8] [9]. However, this method is not without its limitations, primarily manifesting in the following aspects:
   - **Hardware Complexity:** Mapping software functionalities to hardware typically necessitates intricate hardware designs, encompassing specialized processors or circuit modules. This can result in a significant escalation of hardware complexity and costs.
   - **Rigid Design:** Hardware is static, and once the design is finalized, it is challenging to make modifications. This results in lower flexibility within hardwarelization of software, making it difficult to adapt to changing requirements or the addition of new functionalities.
   - **Manufacturing Cost:** Hardware manufacturing costs are relatively high, especially in cases of small-batch production or the need for multiple revisions. This makes hardware design unsuitable for applications that require rapid iteration and cost-effective production.
   - **Adaptability:** Hardware-based systems struggle to adapt to future technological changes. As technology advances, new hardware designs may quickly become obsolete, whereas software is often more amenable to accommodating these changes.
   - **Resource Constraints:** Hardware resources are finite, which can impose limitations on the implementation of complex functionalities in hardware.

Therefore, hardwarelization of software has certain constraints in specific domains, preventing it from achieving the same level of generality as a CPU. Furthermore, due to the semantic boundaries between software and traditional computer architectures, certain semantic deficiencies exist between them.

3) **Challenges of Traditional Computer Architectures:** The fundamental architectural paradigm followed by modern computers is the Von Neumann architecture, characterized by the capability to store both program instructions and data within a single memory space, accessible for processing by the CPU. Furthermore, data within this structure is internally represented and processed in binary form. However, this architecture exhibits a significant performance bottleneck stemming from its heavy reliance on memory, where the speed of information exchange between the CPU and shared memory becomes a primary factor affecting system performance [11]. The enhancement of information exchange speed, in turn, is constrained by various factors, including the speed of memory components, memory performance, and structural considerations. Furthermore, the Von Neumann architecture computers were conceived for Turing computation, which has reached notable speeds and precision in numerical processing. However, their progress in non-numerical processing applications has been comparatively slow. In reality, many phenomena are inherently ambiguous and challenging to express precisely using numerical calculations. Moreover, the occurrence, evolution, and outcomes of events are often unpredictable, making it considerably difficult to automate the processing of such complex tasks using a Von Neumann computer architecture.

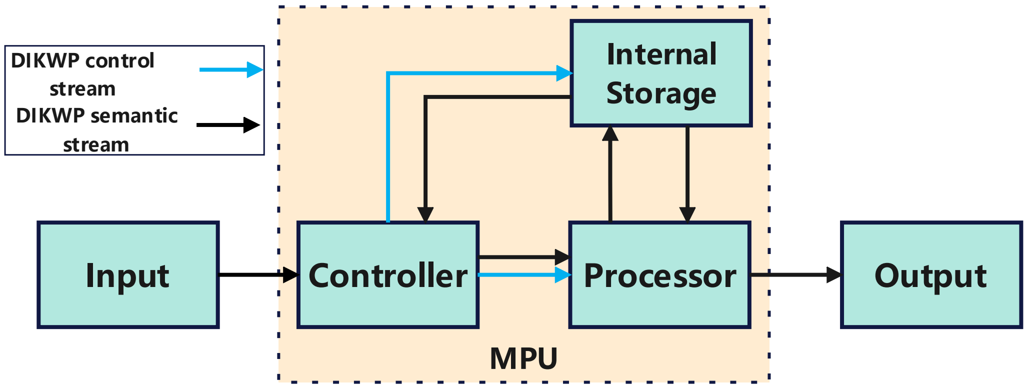


Figure 5-DIKWP computer architecture

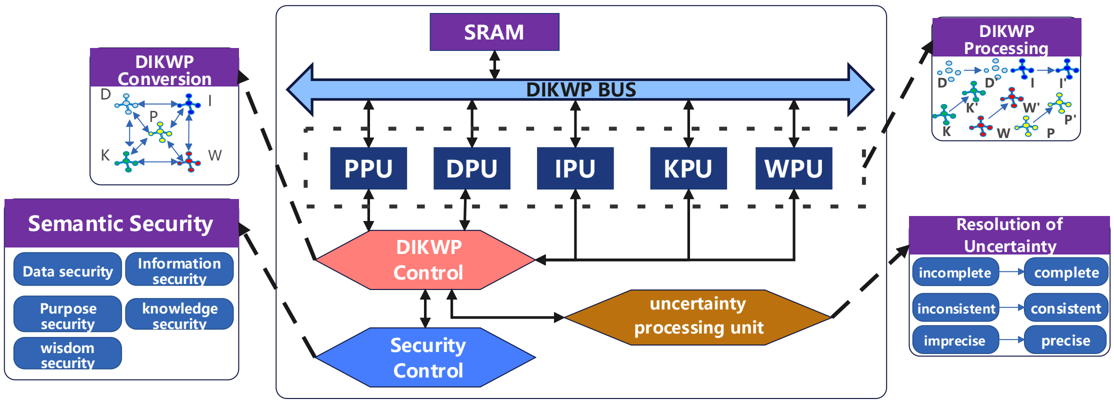


Figure 6-DIKWP hardware architecture

```
Algorithm 1: The procedure of DIKWP semantic security processing
input : $DIKWP'_{SCS}$
output: $DIKWP'_{SCS-S}$
1 if IsSecurityP($DIKWP_{SCS}$) then
2     while $DIKWP_{missing}$ is $\emptyset$ do
3         $DIKWP'_{SCS-S}$ ← Expand($DIKWP_{SCS}$, $SCS_{type}$);
4         $DIKWP_{missing}$ ← UncertaintyF($DIKWP_{SCS}$, $DIKWP'_{SCS-S}$, $SCS_{type}$);
5     end
6 end
7 return $DIKWP'_{SCS-S}$
```

Algorithm 1- The procedure of DIKWP semantic security processing

B. DIKWP on Natural Language Processing

DIKWP model on natural language processing firstly transforms natural language into elements of DIKWP by special processing and then abstracts the elements of DIKWP into graphs. DIKWP graphs are processed and transformed driven by purposes and their logic is verified by formal expressions [2]. But DIKWP model has following problems on natural language processing.

1) **No unified paradigm for conversion:** Some work has been done on processing and conversion between DIKWP content driven by purpose [2] [27]. However, these works currently lack a more unified approach or paradigm for the conversion and mapping of natural language to DIKWP content, as shown in Fig. 3. In other words, we have no defined methodology and process to achieve this transformation through a limited and defined number of steps. If an AC system needs to be constructed, we need to ensure interpretability in implementation and operation, otherwise the AC system will lack interpretability of the processing.

2) **Failing to represent semantic variations:** In the processing of DIKWP model, interactions and conversion between content and cognition have been demonstrated in previous studies [5] [23]. However, they have not adequately reflected the mapping and representation of the processing process to semantics. After constructing the mapping and representation of the semantic space, changes before and after semantic processing can be observed through an observational approach. Therefore, in the existing artificial intelligence systems, when there is a semantic shift or inconsistency during the processing of content, humans cannot directly observe and detect it. It leads to untrustworthiness in the processing process, which contradicts the characteristics of AC systems. In order to establish an AI system that exhibits thinking and behavior consistency, it is imperative to

generate outputs that reflect changes in the semantic space resulting from interactions between cognition and content. This transparency in the system's processing, allowing humans to have visibility into the process, is crucial for building trust.

3) **How the DIKWP model interacts with humans:** The purpose of constructing artificial intelligence systems is to serve humans and interact with them in various ways, such as through natural language text, images, and sound. However, in the processing of DIKWP, everything is abstracted and processed by graph [2] [23] [27]. Graphs are not a convenient semantic carrier for humans, which leads to a semantic gap between the output content after DIKWP -AC processing and human understanding. Due to the lack of reliable and credible processing procedures and related research work in the output and communication of DIKWP -AC, we lack effective theoretical support for human interaction in building DIKWP -AC systems. Therefore, we need to find a way or method to make the DIKWP - AC model interact with humans, allowing for better mutual understanding between humans and the DIKWP -AC system, ultimately serving humans more effectively.

IV. Architecture Design

To seek a solution to the problem described in Section III, we will shift our thinking and try to translate academic difficulties into engineering problems. Building upon the foundation of the DIKWP theoretical model, this section introduces a hardware architecture based on DIKWP, chip design incorporating DIKWP hardwarelization of software, and the concept of software-hardware compatibility, which combines hardware and software elements while emphasizing the semantic security on chip and the DIKWP semantic processing.

A. DIKWP Computer Architecture

This section attempts to alleviate the problems in Section III in terms of engineering but faces many challenges. we propose a framework for DIKWP-based computer architecture and chip design. In order to alleviate the problem of interaction between DIKWP models and humans, this section also proposes an internal DIKWP language, which relies on the DIKWP runtime environment. The underlying layer of the DIKWP language is based on the DIKWP semantic codec, which can encode the content of the DIKWP language into DIKWP hardware instructions or decoding and restoring the hardware outputs, and the primary logical processes is on the left side of Figure. 4.

B. DIKWP Runtime Environment

To address the challenges and difficulties posed by traditional programming languages, we have introduced a framework model for the DIKWP -based runtime environment, based on the DIKWP model theory. The model encompasses the software runtime environment under the DIKWP hardware architecture and outlines the general processing flow for human-computer interaction. On the left side of Fig. 4, users engage in interaction with DIKWP software through natural language. The DIKWP software platform, in turn, processes this natural language using a DIKWP language interpreter to convert it into DIKWP language. The DIKWP language, serving as a bridge between humans and computers, offers a relatively precise, consistent, and comprehensive medium for human-computer interaction, thereby minimizing semantic loss in these interactions as much as possible.

To facilitate bidirectional communication between the DIKWP language and DIKWP hardware, the DIKWP runtime environment also necessitates a DIKWP semantic codec. This codec is responsible for transforming the DIKWP language into a DIKWP semantic stream. Semantic streams are categorized into five distinct types, namely, data semantic stream, information semantic stream, knowledge semantic stream, wisdom semantic stream, and purpose semantic stream, each with its own processing methodology. After receiving and processing the semantic stream, it can be represented in the form of a semantic graph within the DIKWP semantic space. The processed semantic stream, following the DIKWP semantic codec's operation, represents semantic content before any purpose-based manipulation, and the internal transformation process unfolds as follows:

- The first step involves receiving the input DIKWP semantic stream and transforming it into a DIKWP semantic graph. Subsequently, the semantic orientation of the DIKWP graph is modified, aligning it with the purpose, which is achieved through a purpose alignment process.
- The converted DIKWP semantics are then examined, and the necessary DIKWP semantic content for scenarios that meet purpose is generated.
- The missing DIKWP semantic content is treated as a new purpose with elevated priority. Subsequently, the content is merged and output as a new DIKWP semantic stream.

The semantic stream, after passing through the purpose-driven processing module, contains DIKWP semantic content with semantic tendencies. It then proceeds to the resource allocation module. The resource mapping module allocates the DIKWP semantic stream to the corresponding hardware or processing clusters to ensure efficiency when executing various tasks and functions. It serves as the communication bridge between software and hardware in the DIKWP architecture.

C. DIKWP Language

To address the challenges in processing knowledge, wisdom, and purpose using traditional programming languages, we proposed the DIKWP language. As a domain-specific language (DSL), it is designed for human-computer interaction within the DIKWP architecture. The DIKWP language serves as a means of expressing data, information, knowledge, wisdom and purpose, with example provided in Table 2.

In Table II, various elements can be expressed in different forms through the DIKWP language. These representations can be obtained within the computer through the process of either writing or transformation, resulting in diverse forms of data, information, knowledge, wisdom and purpose.

- **Data:** Data refers to discrete, semantically ambiguous content, such as numbers, characters, or their collections.
- **Information:** Information has a certain semantic scope, responsible for defining and organizing data to make it meaningful. It has two forms of representation: one is the definition of data, and the other is by defining a function or operation-like form to fill in data or information.
- **Knowledge:** Knowledge is represented through a combination of facts, information, and logical expressions. It is the presence of logical expressions that enables the system to extract interpretable knowledge rules and form the content from the knowledge.
- **Wisdom:** Wisdom, fundamentally, is information manifested in its highest semantic value within specific dimensions. For instance, in Table II, the task is to identify the maximum element within the array ”aa” and assign it to ”ArrayMax,” while ”Where” signifies the value path that, within a specific set of rules, leads to this objective.
- **Purpose:** Purpose is the driving force behind *DIKWP*, representing the execution of purpose functions in semantic terms. It both takes input and produces output in the form of *DIKWP* elements.

To ensure that the *DIKWP* language can be interpreted for corresponding instructions in downstream tasks, it includes specialized keywords, in addition to the conventional mathematical and logical operators, which are used for execution and defining semantic boundaries, as shown in Table 3.

D. *DIKWP* Hardwarelization of Software

The main purpose of hardwarelization of software is to shift the implementation of certain software functions to hardware and integrate them onto chips. It is done in order to execute repetitive, time-consuming, and inflexible functions more efficiently. The *DIKWP*-AC chip represents an innovative computer architecture designed to achieve AI systems with consciousness. Its core concept places the processing of data, information, knowledge, wisdom, and purpose at the center, aiming to simulate human cognitive processes. The *DIKWP* microarchitecture is depicted in Figure 6, and this part will provide a design and description of different hardware module functionalities.

1). Processor: The microarchitecture in the *DIKWP* architecture is mainly a processor that contains data, information, knowledge, wisdom, and purpose, but in terms of the functional division of processing, since the processing of low level functions is easier to implement as compared to the processing of high-level functions, the data processors in the *DIKWP* operator can be abstracted to existing general-purpose processors, i.e., the Turing computations that they are capable of performing on the data. And information processors are the higher-level operations that can be performed on specific semantic data streams, such as encoding and decoding of general video streams and audio streams, complex matrix operations, hash coding, and other operations. Whereas knowledge, wisdom and purpose are processed as high-level abstractions, the current implementation of their based processes at the circuit level needs to be followed by deeper research and practice.

- **Data Processing Unit (DPU):** DPU is capable of simple computations that can be performed on data, such as arithmetic operations, logical operations, displacement operations, and other operations. In addition, the DPU transforms the input data into a machine-understandable form while searching for common semantics in it to standardize it into a single concept, a process that involves techniques such as data cleaning, feature extraction and pattern recognition.
- **Information Processing Unit (IPU):** IPU provide higher level and more abstract computations than DPU, such as encoding, decoding, compression, and other operations, similar to the complex instruction's architecture. The responsibility of its IPU is to classify, categories and organize the input information to form a higher level of understanding.
- **Knowledge Processing Unit (KPU):** The task of a KPU is to construct complete concepts or patterns through observation, learning, and abstraction. the KPU builds a deep understanding of a particular domain or issue. This knowledge forms the cognitive foundation of the chip and provides support for more advanced intelligence and decision making. But currently, stand-alone KPU cannot exist and process knowledge content on their own. KPU needs to build understanding and interpretation of the world through other levels of content (*DIKWP*).
- **Wisdom Processing Unit (WPU):** The essence of the function of the WPU is information, which reflects the results of the judgement of the value system. The WPU incorporates individual, ethical and moral factors into the decision-making process while considering multiple aspects such as feasibility and sustainability. However, the current hardware design for decision making for the fusion of individual cognition and objective rules needs to be investigated more deeply in subsequent studies.
- **Purpose Processing Unit (PPU):** PPU provides a fine grained reconfigurable circuit framework that allows the content of external inputs to be constructed from the content of the purpose and executed as the corresponding circuit code. The input to a PPU can be abstracted into a function structure and corresponding parameters that need to be input, and the result of the processing is the output of the function execution.

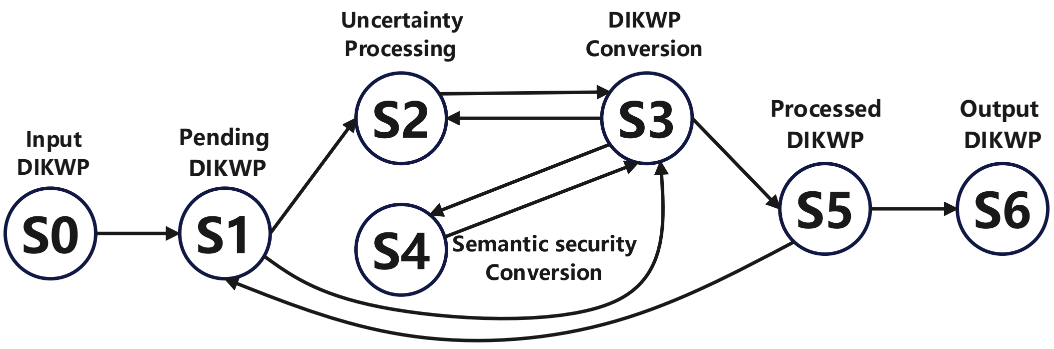


图 7-Finite state machine of *DIKWP* control unit

2). *DIKWP* Control Unit: The ability of the *DIKWP*- AC chip to simulate human cognitive processes hinges on its efficient transformation and processing of DIKWP resources. The DIKWP Control Unit encompasses the logic for chip circuit control, which includes logical judgment and flow control of input content. Furthermore, the Control Unit is responsible for managing the transformation process of semantic stream from data, information, knowledge, wisdom and purpose units, ensuring smooth transmission and conversion between these elements to meet the requirements of various modules and processing units. The Finite State Machine (FSM) of the *DIKWP* Control Unit, as shown in Figure 7, illustrates different.

- S0 represents the initial input of semantic stream, denoting unprocessed resources.
- S1 state corresponds to the status of parsing and allocation processing units, analogous to the decode instruction in CPU.
- S2 represents the state of uncertainty processing, where the process involves handling uncertainty in data, information, knowledge, wisdom, or purpose within the *DIKWP*, aiming to minimize the gap between the actual outcome and the expected outcome under uncertain input conditions.
- S3 is the transformation processing state of *DIKWP*, which includes intra-type and cross-type transformations and compensations driven by purpose.
- S4 signifies the state of semantic security processing, where semantic security conversions operate when *DIKWP* includes secure purposes. The conversion principle is to expand the semantic distance between non-stakeholders and the current *DIKWP* without disrupting the original semantics.
- S5 indicates the status upon completion of the current phase, returning to the control unit. The control unit evaluates whether the processed content should be output or continued for further processing. If further processing is required, it returns to S1 for reallocation to processing units; otherwise, the processing result is output.
- S6 represents the completed state, with results output through the *DIKWP* Control Unit.

Due to the characteristics of data flow processing, as shown in Figure 7, the control unit can only manage the internal scheduling of the processing unit but cannot handle global situations. Suppose we have two tasks (Task A and Task B), with Task A having longer execution time, and Task B having a shorter time but depending on the processing results of Task A. This situation may cause the processing unit for Task B to be in a paused state for some time. To mitigate such situations, we require an efficient top-level scheduling algorithm, such as out-of-order execution, to enhance the processor's throughput. However, further research is needed to adapt the optimal scheduling algorithm effectively.

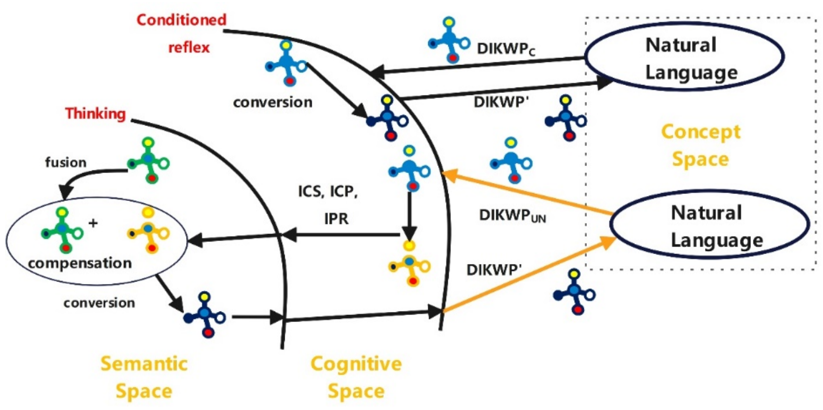


Figure 8- Uncertainty Processing in *DIKWP*-AC

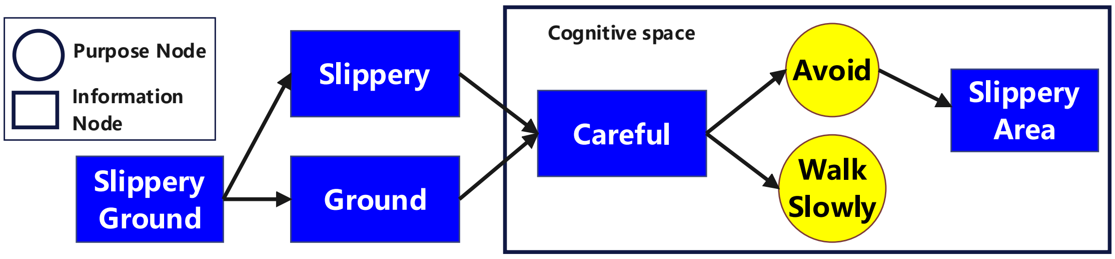


Figure 9-A case of warning text

3). The Uncertainty Processing Unit (UPU): In Figure 8, regarding the handling of uncertainty, we need to model it based on human reflex and processing of natural language content. We abstract human language processing into two levels: one is the second-signal system processing based on conditioned reflexes, represented by the arcs in Figure 8, which we define as surface-level processing. This surface-level processing is defined as the Cognitive Space on *DIKWP*. This type of surface processing can be abstracted as the direct reflection of specific semantics in natural language by the AC system. These reflexes are acquired through postnatal training and communication.

We define the process of direct reflection as a function, denoted as $f_{cf}(DIKWP_{CT})$, which represents the corresponding results of mapping *DIKWP*'s content directly to the cognitive and semantic space:

$$DIKWP_{OUT} = f_{cf}(DIKWP_{CT}) \tag{8}$$

For example, as depicted in Figure 9, when humans see a warning sign that reads "Slippery ground", they naturally become alert. Another factor is that since the input language content is incomplete, inconsistent, or imprecise, no specific reflexes are formed in the Cognitive Space, or multiple reflexes occur, resulting in either a null set or a set containing multiple *DIKWP* elements:

$$f_{cf}(DIKWP_{UN}) = \begin{cases} \emptyset \\ \{DIKWP_{un1}, DIKWP_{un2}, \dots\} \end{cases} \tag{9}$$

According to equation 10, we define the thought function $T(DIKWP_{UN})$, which encompasses two processing steps. One step involves compensating for the uncertainty in the output of $f_{cf} = (DIKWP_{CT})$, and the other involves the fusion and conversion of *DIKWP* elements under the influence of purpose-driven:

$$\{DIKWP_{UN1}, DIKWP_{UN2}, \dots\} = T\left(f_{cf}(DIKWP_{UN})\right) \tag{10}$$

For instance, humans exhibit conditioned reflexes when faced with mathematical problems, automatically generating simple calculation methods in their minds. However, more complex computations require deeper thinking, a process that occurs in the semantic space. Through the uncertainty handling of *DIKWP*, conceptual content is mapped to cognition, completing the transformation from conceptual space to cognitive space in the semantic space. Through specific training, even deep-level thinking and reasoning can become conditioned reflexes. For example, after systematic mathematical training, multiplication tables and derivative formulas become conditioned reflexes, eliminating the need for deeper reasoning and calculation. Therefore, the UPU, acting as the central controller for uncertainty handling, is responsible for processing incomplete, inconsistent, or imprecise inputs. It then utilizes existing *DIKWP* elements for evolution and inference, ultimately producing results. This process is a collaborative operation involving the UPU, 𝒟𝐼𝒦𝒲𝒫 control units, and 𝒟𝐼𝒦𝒲𝒫 processing units.

4). The Security Control Unit (SCU): The SCU primarily handles whether the output of the transformation process under the influence of purpose fulfills its security purpose. The hardware implementation of the SCU aims to prevent code tampering, thereby ensuring computational security, reliability, and efficiency. The primary function of the SCU is to guarantee semantic security for data, information, knowledge, wisdom, and purpose. It accomplishes this by establishing relationships that extend beyond closely related elements. The primary processing procedure of SCU is illustrated in Algorithm 1. When there is a secure purpose in the semantic stream, the SCU will expand the input semantic stream. The SCU computes the difference in the 𝒟𝐼𝒦𝒲𝒫 semantic space before and after the expansion through uncertainty processing, denoted as $DIKWP_{missing}$ in the algorithm. If $DIKWP_{missing}$ is not an empty set, the expanded 𝒟𝐼𝒦𝒲𝒫 semantic stream is output as 𝒟𝐼𝒦𝒲𝒫 语义流输出为$DIKWP_{SCS-S}$。 For example, in the 𝒟𝐼𝒦𝒲𝒫 language, an information is represented as follows:

$$I_e = Met(Classmates(Names = \{'Lee','Wang','Sun'\}), Location('Haikou')) \quad 11$$

In equation 11, I_e originates from the natural language statement: "Li, Wang, and Zhang are classmates; they meet in Haikou." If these three individuals do not wish others to know their exact meeting location, but still need to include the semantic information "Haikou" in their communication, the result after processing by the SCU is as follows:

$$I'_e = Met(Classmates(Names = \{'Lee','Wang','Sun'\}), Location('Hainan')) \quad 12$$

In equation 12, the semantics of "Hainan" include "Haikou," but in their cognitive space, it refers to Hainan. After processing by the SCU, their primary intent is not directly understood by third parties, thereby expanding the semantic space and increasing the difficulty of searching for the location. The 𝒟𝐼𝒦𝒲𝒫-AC platform alleviates some of the challenges faced when implementing 𝒟𝐼𝒦𝒲𝒫 with traditional programming languages and narrows the cognitive gap in human-machine interactions. In future work, we will delve into the specific details of chip implementation and evaluate the adaptability of the software platform. Our proposed 𝒟𝐼𝒦𝒲𝒫 integrated hardware-software platform will make the construction and operation of 𝒟𝐼𝒦𝒲𝒫-AC systems more convenient, reliable, and efficient.

V. Conclusion

Our research, based on the 𝒟𝐼𝒦𝒲𝒫-AC theory, has analyzed the current constraints and challenges in the implementation of 𝒟𝐼𝒦𝒲𝒫-AC systems in both hardware and software. Consequently, we propose the 𝒟𝐼𝒦𝒲𝒫 hardware architecture and the integrated 𝒟𝐼𝒦𝒲𝒫 chip encompassing storage, computation, and transmission. Within the 𝒟𝐼𝒦𝒲𝒫 chip, we introduce the microarchitecture of 𝒟𝐼𝒦𝒲𝒫 processing unit, UPU, and SCU based on 𝒟𝐼𝒦𝒲𝒫 semantic communication. Furthermore, to enhance the versatility of the 𝒟𝐼𝒦𝒲𝒫 chip, we present the 𝒟𝐼𝒦𝒲𝒫 language and its associated runtime environment, creating a software and hardware ecosystem for 𝒟𝐼𝒦𝒲𝒫. This platform, to some extent, mitigates the challenges associated with implementing *DIKWP* using traditional programming languages and reduces the cognitive gap in human-computer interaction. In future work, we will delve into the specific details of chip implementation and assess the adaptability of the software platform. We present *DIKWP* software-hardware integrated platform will make the construction and operation of *DIKWP* -AC systems more accessible, reliable, and efficient.

Refrence:

[1] C. Thon, B. Finke, A. Kwade, et al. Artificial intelligence in process engineering. Advanced Intelligent Systems, vol. 3, no. 6, pp. 2000261, 2021.
[2] W. Liang, G. A. Tadesse, D. Ho, et al. Advances, challenges and opportunities in creating data for trustworthy ai. Nature Machine Intelligence, vol. 4, no. 8, pp. 669–677, 2022.
[3] Y. Duan, Beyond Attention: Attention is NOT all you need. September 2023, DOI:10.13140/RG.2.2.10379.05920.
[4] Y. Duan. Relativity of Consciousness and DIKWP. August 2023, DOI:10.13140/RG.2.2.36142.89922.
[5] Y. Duan, L. Shao, G. Hu, et al. Specifying architecture of knowledge graph with data graph, information graph, knowledge graph and wisdom graph. SERA 2017, pp. 327–332, IEEE, 2017.
[6] Y. Duan, Trans-modal, trans-scale, and meso-scale subjective cognitive semantic modeling and analysis for data, information, and knowledge overload, April 2020, DOI:10.13140/RG.2.2.31481.01125.
[7] Y. Huang and Y. Duan. Towards purpose driven content interaction modeling and processing based on dikw. SERVICES 2021, pp. 27–32, IEEE, 2021.
[8] S. Ambrogio, P. Narayanan, A. Okazaki, et al. An analog-ai chip for energyefficient speech recognition and transcription. Nature, vol. 620, no. 7975, pp. 768775, 2023.
[9] J. Barba, M. J. Santofimia, J. Dondo, et al. Fpga acceleration of semantic tree reasoning algorithms. Journal of Systems Architecture, vol. 61, no. 3-4, pp. 185196, 2015.
[10] Y. Chen, M. Nazhamaiti, H. Xu, et al. All-analog photoelectronic chip for highspeed vision tasks. Nature, 2023, DOI:10.1038/s41586-023-06558-8.
[11] H. Thimbleby. Modes, wysiwyg and the von neumann bottleneck. IEE Colloquium on Formal Methods and Human-Computer Interaction: II, pp. 4–1, IET, 1988.
[12] Migliato Marega G, Zhao Y, Avsar A, et al. Logic-in-memory based on an atomically thin semiconductor. Nature, 587(7832): 72-77, 2020.
[13] Y. Duan, L. Shao, and G. Hu. Specifying knowledge graph with data graph, information graph, knowledge graph, and wisdom graph. IJSI 2018, vol. 6, no. 2, pp. 10–25, 2018.
[14] Y. Duan. Existence Computation and Reasoning (EXCR) and Essence Computation and Reasoning (ESCR) based revelation of the semantics of point, line and plane. February 2022, DOI: 10.13140/RG.2.2.32383.89767.
[15] Y. Duan, X. Sun, H. Che, et al. Modeling Data, Information and Knowledge for Security Protection of Hybrid IoT and Edge Resources. IEEE Access, vol. 7, pp. 99161-

99176, 2019.
[16] H. Gao, Y. Duan, L. Shao. et al. Transformation-based processing of typed resources for multimedia sources in the IoT environment. Wireless Netw 27, 33773393 2021.
[17] Y. Li, Y. Duan, Z. Maamar, et al. Swarm differential privacy for purpose-driven data-information-knowledge-wisdom architecture. Mobile Information Systems, vol. 2021, pp. 1–15, 2021.
[18] Y. Huang, Y. Duan. Fairness Modelling, Checking and Adjustment for Purpose Driven Content Filling over DIKW. HPCC/DSS/SmartCity/DependSys 2021, pp. 2316-2321, IEEE, 2021.
[19] T. Hu and Y. Duan. Modeling and measuring for emotion communication based on dikw. SERVICES 2021, pp. 21-26, IEEE, 2021.
[20] Y. Duan, V.T. Pham, M. Song, et al. Ultimate of Digital Economy: From Asymmetric Data Economy to Symmetric Knowledge and Wisdom Economy. SoMeT, pp. 85-96, 2023.
[21] Y. Mei, Y. Duan, L. Yu, et al. Purpose Driven Biological Lawsuit Modeling and Analysis Based on DIKWP. CollaborateCom 2022, pp. 250-267, Springer, 2022.
[22] Z. Guo, Y. Duan, L. Chen, et al. Purpose Driven DIKW Modeling and Analysis of Meteorology and Depression. HPCC/DSS/SmartCity/DependSys 2022, pp. 21262133, IEEE, 2022.
[23] Y. Mei, Y. Duan, L. Chen, et al. Purpose Driven Disputation Modeling, Analysis and Resolution Based on DIKWP Graphs. HPCC/DSS/SmartCity/DependSys 2022, pp. 2118-2125, IEEE, 2022.
[24] Y. Duan. The Operating Principles of the DIKWP Artificial Consciousness Chip. September 2023, DOI: 10.13140/RG.2.2.24718.33602.
[25] Y. Duan. DIKWP Chip vs. Pulse Chip vs. Quantum Chip: The Future Path of Artificial Consciousness Computing. September 2023, DOI: 10.13140/RG.2.2.27234.91846.
[26] B.M. Lake, M. Baroni, Human-like systematic generalization through a metalearning neural network. Nature, 2023, DOI: 10.1038/s41586-023-06668-3.
[27] Y. Wang, Y. Duan, M. Wang, et al. Resource Adjustment Processing on the DIKWP Artificial Consciousness Diagnostic System. DIKW 2023, 2023.
[28] Y. Duan. DIKWP-AC Artificial Consciousness: Fusing Physiology and Mathematics. July 2023, DOI:10.13140/RG.2.2.26720.87040.

翻译：DIKWP团队吴坤光、段玉聪

意图驱动的数据、信息、知识、智慧融合发明创造方法：DIKWP-TRIZ

段玉聪教授

- DIKWP-AC 人工意识（全球）团队发起人
- AGI-AIGC-GPT 评测 DIKWP（全球）实验室创办者
- 世界人工意识大会发起人（ArtificialConsciousness2023,AC2023,AC2024)
- 国际数据、信息、知识、智慧大会发起人（IEEEDIKW2021、2022、2023）
- 斯坦福全球顶尖科学家“终身科学影响力排行榜”（海南信息技术）唯一入选
- 海南人工智能技术发明领域唯一全国奖（吴文俊人工智能奖）获得者
- 中国创新方法大赛总决赛（海南代表队）最好记录保持者
- 海南省发明专利（信息技术领域）授权量最多者
- 全国企业创新增效大赛海南最好成绩保持者
- 全国人工智能应用场景创新挑战赛总决赛海南最好成绩保持者
- 海南唯一入选“首届科技期刊高质量发展大会 100 篇”
- 海南省最美科技工作者（并入选全国候选人）
- 首届中国“AI+”创新创业大赛最佳创意奖