BitAgere: A multi-dimensional Agere interconnection system based on Bitcoin

Abstract

In multiple decentralized mechanical consensus environments, how to build a trustworthy consensus field is the core challenge currently faced. This paper draws on the feedback loop concept from cybernetics and proposes a three-element closed-loop consensus model called "Control-Compute-Communicate" (C-K-M), and based on this, constructs a multi-consensus fusion system with adaptive characteristics called BitAgere. This system is built on the security and decentralization of Bitcoin, using mechanical contracts to connect multiple agents to form the Agere system, and through the Agere consensus mechanism, it creates a scalable consensus field, enabling interaction and cross-domain collaboration among various types of agent systems. The Agere consensus mechanism can promote trust and collaboration among multiple agents under conditions of decentralized management, providing a scalable security system solution for multi-agent cooperation in a decentralized environment.

Keywords: mechanical consensus, consensus field, Cognito theory, Agere system, mechanical contract, Agere consensus, cross-domain

1. Introduction

Satoshi Nakamoto's consensus achieved ledger consistency and trustworthy transactions in a decentralized environment, providing an important practical case for peer-to-peer electronic cash systems. However, its mechanism is limited to the single dimension of ASIC non-general-purpose computing power and Coin, which are mutually mapped: ASIC non-general-purpose computing power only serves the generation and transaction verification of Coin, while the value of Coin, in turn, reinforces the specificity of ASIC computing power, thus forming a closed-loop system. To address this issue, blockchain introduced a Turing-complete VM, but

fell into the trap of focusing only on the computation at the VM layer, without perceiving the consensus layer of external entities.

This article aims to address: how to design an adaptive consensus mechanism that can perceive the external environment and integrate a series of independent mechanical consensus systems into a unified Agent consensus field, ultimately achieving the integration of Crypto and Agent.

2. Cognito theoretical model

2.1 From computability and incompleteness to the demand for meta-rules

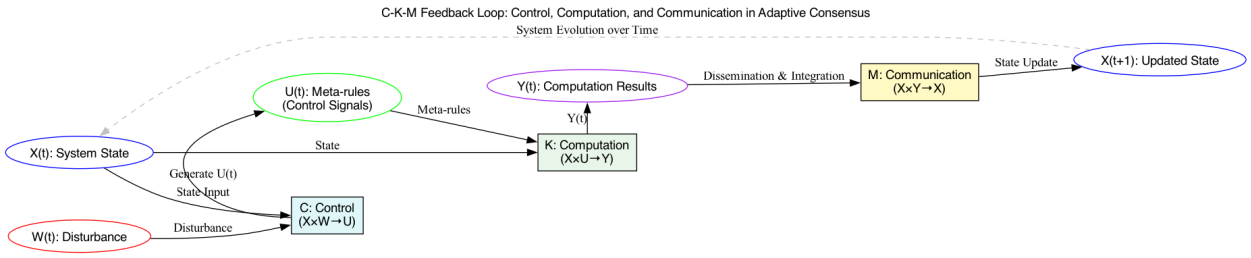
The theory of Turing computability indicates that computability is bounded by the scope of predefined rules; if the environment exceeds the initial design domain, the system struggles to respond in a timely manner. Gödel's incompleteness theorem suggests that any closed system has "blind spots" that cannot be self-proven within its own framework. Analogous to consensus mechanisms, when new types of attacks and strategy combinations emerge, pure computational logic cannot self-expand or improve. To break through this inherent limitation, a layer of meta-rules is needed to dynamically correct and adapt the consensus protocol in order to cope with unforeseen changes.

2.2 Introduction of Wiener Cybernetics and Feedback Closed Loop

The concept of feedback control:

Cybernetics (Wiener, 1948) provides a foundational methodology for maintaining system target output in uncertain environments: by comparing the system's current output with the expected target, the error is fed back to the control layer, which dynamically adjusts system parameters and strategies. This feedback loop allows the system to maintain a steady state or continue evolving despite disturbances.

Three-dimensional closed-loop framework:



Referring to cybernetic ideas, we abstract the distributed consensus system into a closed-loop structure composed of three parts: Control(C), Computation(K) and Communication(M). Define the following function mapping:

- $C : X \times W \to U$: The control layer extracts information from state $x(t)$ and environmental disturbance $w(t)$, outputting meta-rule instruction $u(t)$. This corresponds to the strategic layer of the system, dynamically tuning consensus protocol parameters (such as difficulty targets and economic incentive mechanisms).
- $K : X \times U \to Y$: The calculation layer performs specific verification and accounting operations (such as transaction verification and block construction) on the state $x(t)$ under the control instruction $u(t)$, outputting the result $y(t)$.
- $M : X \times Y \to X$: The communication layer spreads and integrates the calculation result $y(t)$ across the network, allowing all nodes in the network to reach a consensus update on the next state $x(t+1)$.

The system state evolves over time through *C* - *K* - *M* loop iterations:

$$x(t) \to u(t) \to y(t) \to x(t+1)$$

When the external disturbance *w(t)* changes, the control layer C can adjust *u(t)* in real time, thereby affecting the computation and communication process, allowing the system to respond adaptively to new problems under feedback closed-loop.

2.3 Establishment of the cybernetic triad model (Cognito theory)

Summarizing the above analysis, we arrive at a cognitive leap: a control layer (meta-rule layer) needs to be introduced on top of the purely computational model

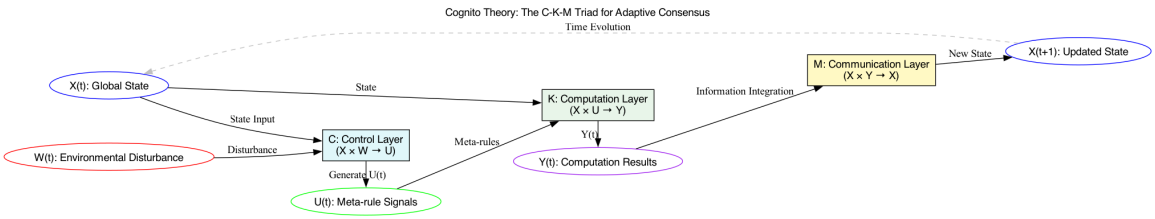
to integrate with communication and build a consensus framework with adaptive evolutionary capabilities. This idea can be formalized at the mathematical and logical level as follows

Cybernetic triad model (Cognito theory):

Definition:

A distributed consensus system S is represented as a triple (C,K,M) that satisfies:

1. $C : X \times W \to U$ provides the ability to regulate the source rules and respond dynamically to environmental disturbances.
2. $K : X \times U \to Y$ Executes consensus logic and computation operations under the given strategy.
3. $M: X \times Y \to X$ ensures that information is effectively transmitted and integrated across the entire network, achieving status updates.



Through this model, we can analyze the stability, adaptability, and scalability of the system under uncertain conditions. If we further utilize stability determination methods from control theory (such as Lyapunov analysis, H∞ control, or robust control theory), it is expected that we can mathematically rigorously prove the conditions under which the system achieves steady state or continuous evolution under a certain set of disturbances.

3. Cognito Theory and Unified Agent Markets

By comparing Bitcoin and Ethereum, this chapter points out that Bitcoin can utilize Satoshi Nakamoto's consensus to connect to external arithmetic power, while Ethereum lacks the ability to perceive the real environment although it has improved the computational power in the computing layer. It is further concluded

that consensus is more important than computation in multi-party collaboration in a distributed environment. By observing that Crypto and AI have the common basic unit of Agent through Cognito theory, we introduced the common abstraction of Agent to unify Crypto and AI, and proposed BitAgere and the concept of "Consensus Field" in order to integrate Crypto and AI with Agent. In order to integrate Crypto and AI with Agent, BitAgere and the concept of "consensus field" were proposed to integrate multi-chain, multi-system, and multi-agent into a unified ecosystem that can be interoperated across domains.

3.1 Bitcoin: the consensus layer's perception of the outside world

From the perspective of the Cognito theory of "control, computation, and communication" (C-K-M), the outstanding significance of Bitcoin lies in the close coupling with the real world through the direct connection of the consensus layer to external arithmetic power. Its core structure includes:

- Control layer
    - Difficulty Retargeting: Every 2016 blocks, the mining difficulty is automatically adjusted upward or downward according to the block rate of the whole network, forming a dynamic closed-loop feedback on the environment (arithmetic power).
    - Halving of block incentives: discrete "convergence" of block subsidies at predetermined block heights to balance inflation and incentives.
- Computation layer
    - SHA-256 PoW algorithm: miners have bookkeeping rights only if they win the hash race.
    - Script validation mechanism: nodes perform script validation (e.g. OP__CHECKSIG) for each transaction within the block, which jointly maintains the consistency of the UTXO set and ensures the validity of transactions.
- Communication layer
    - P2P Gossip Network: blocks and transactions are propagated between nodes through decentralization, complemented by structures such as the

Merkle Root in the block header to guarantee the integrity of the ledger.

- Data synchronization mechanism: new nodes can quickly join the network through block header verification and data synchronization to ensure network scalability.

Bitcoin has successfully constructed a centerless and trusted distributed ledger by directly coupling external physical work (power, arithmetic) with on-chain security through the Satoshi Nakamoto consensus. This design combines engineering implementation with adaptive control, providing important insights for the evolution of subsequent consensus algorithms.

3.2 Ethereum: Breakthroughs in the Computing Layer and Limitations in the Consensus Layer

Ethereum's introduction of Turing-complete EVMs on top of Bitcoin upgraded the compute layer from simple money transfers to Turing-complete Virtual Machines (EVMs), but at the same time exposed serious limitations:

- Control layer:
  - Lack of perception: The perception of EVM is completely separated from the underlying consensus, and there is no attempt to perceive the external world through the consensus layer, which ultimately leads to the isolation of the computational layer of EVM.
- Computation layer
  - Computational isolation: contract execution relies entirely on on-chain inputs and lacks the ability to perceive the external world. tokens in DeFi, although logically meaningful on the chain, lack direct binding to physical resources and actual behavior, making it difficult for on-chain activities to affect the real world.

3.3 Agent: a breakthrough in unified abstraction

The analysis of Bitcoin and Ethereum shows that Bitcoin realizes adaptive sensing of external arithmetic resources through Satoshi Nakamoto's consensus mechanism, but this sensing mechanism is limited to a single dimension. In order to expand the dimension of interaction between blockchain and the real world, we

study the most dynamic AI field, and find that Crypto and AI have an essential commonality: the Agent.

- **Definition and commonality of Agent**: An Agent can be defined as any entity that can autonomously execute code and perform input and output. Under this definition:
    - Validators in Bitcoin and Ethereum are Agents, which are connected to each other through mechanical contracts (consensus rules) to form a decentralized blockchain network with mechanical consensus.
    - AI large model as Agent is to regulate its code of behavior through preset training objectives (e.g. loss function), on the basis of which it can autonomously process inputs and generate outputs, showing reasoning and decision-making capabilities, and at the same time can interact with external systems through APIs or other interfaces, thus forming a service network.
- **Core Insight on Consensus over Computation**: From our analysis of Bitcoin, we have come to the important conclusion that consensus is more important than computation in the convergence of Crypto and AI. This is because:
    - Bitcoin has proven that effective connectivity to the outside world can only be achieved by extending the consensus layer.
    - Computational power increases (e.g. EVMs) do not solve the problem of severance from the real world.

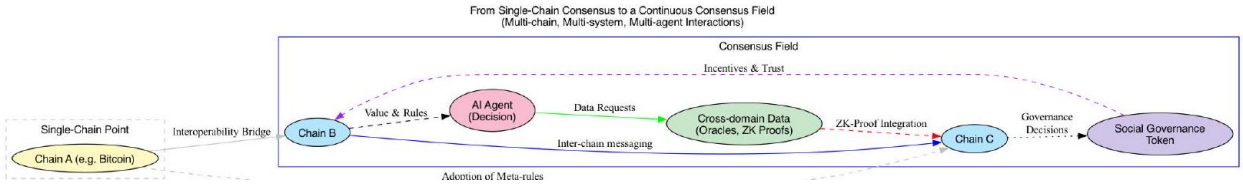
3.4 BitAgere: Toward a Unified Agent Market and Cross-Domain Scaling

In the previous section, we analyzed Crypto such as Bitcoin, Ethereum, and Agent concepts based on the C-K-M structure, and found that most of the existing solutions are limited to a single consensus or computation model, and it is difficult to realize full interaction with external environment and multiple Agents. To this end, we propose the **BitAgere** system, whose core concept is to expand the consensus assetization from the one-dimensional arithmetic mapping of Bitcoin to a multi-Agent ecosystem, so that different types of Agents can evolve collaboratively under unified incentives and rules.

In terms of specific design, BitAgere contains two key concepts:

- Agere system: an autonomous network composed of multiple Agents through a mechanical contract, each Agent strictly follows the consensus rules to complete the interaction and bookkeeping, forming a decentralized decision-making and execution of the closed loop.
- Consensus Field: On top of this, multiple Agere systems are connected to each other through the consensus assetization mechanism to gradually build a unified Agent market. Through the Consensus Assetization Mechanism, each Agere system can map its own credit or value into the Consensus Field, and achieve cross-domain interoperability with Agents in other networks.

Currently, different Crypto, Agents or applications are often independent of each other as "islands". BitAgere treats single-chain consensus as a point, and extends multi-chain, multi-system, multi-agent consensus to a relatively continuous network structure through the concept of "consensus field". Through the concept of "consensus field", BitAgere extends the consensus of multi-chain, multi-system and multi-agent to relatively continuous network structure. The concept of "consensus field" extends multi-chain, multi-system, multi-agent consensus to a relatively continuous network structure:



- **C-K-M Expansion**: in the consensus field, the control layer is no longer limited to single coefficient adjustment, but cross-domain fusion of rules from multi-dimensional subjects (human society, other blockchains, AI Agents); the computation layer integrates multi-source validation logic (cross-chain message authentication, zero-knowledge proofs, AI Decision Mapping); the communication layer extends to cross-domain routing, cross-system relay protocols, enabling information and credit to flow globally as a continuous field.
- **Further sublimation of consensus assetization**: in the consensus field, assets are not only UTXO-style single-chain value units, but also include cross-chain

credit certificates, social governance tokens, parameterized incentives for AI models, and other multi-level value abstractions. Consensus assets become the carrier for transferring trust in the consensus field, enabling multi-dimensional and continuous interaction of trust among humans, machines, contracts and Agents.

Based on the above design, BitAgere provides a unified and scalable framework for multi-chain interoperability and multi-agent collaboration.

4. BitAgere Multi-Agent Consensus Framework

This section proposes the BitAgere Multi-Agent Consensus Framework, which aims to inherit the advantages of Bitcoin's decentralized security and value transmission, and at the same time, draws on Cognito's theory to provide an adaptive solution for the scenario of "AI Agent + Crypto". The framework hopes to build a centerless autonomous ecosystem, so that multi-agents can interact deeply and play dynamically between Crypto and AI.

4.1 Framework overview and design motivation

4.1.1 Existing limitations and problems

1. **Single-dimensional perception**

   Bitcoin can only sense and incentivize external arithmetic with Proof of Work (PoW), and is not able to directly incorporate a wider range of external resources or agent behaviors (e.g., AI arithmetic, cross-chain data input, etc.) into the consensus process. This makes it difficult to further expand to collaborative applications in multiple scenarios.

2. **Inadequate perception of the external environment**

   Although Turing-complete chains such as EtherChannel have strengthened the computational layer, they still lack a strong mechanism to directly sense and integrate external entities (e.g., AI systems or physical environments), resulting in a disconnect between the computational layer and external interactions, making it difficult to truly form a closed loop.

3. **Lack of incentives for multi-agent collaboration**

   When Agents with different functions and attributes are co-located in the same network, if there is no unified synergistic mechanism and incentive rules, it is difficult to ensure that they can form an effective cooperation or reasonable competition under the decentralized environment, which restricts the potential value of the multi-agent system.

4.1.2 Objectives of the chapter

This chapter describes the key design concepts and overall framework of BitAgere, centered on the above issues. Specifically, the goals of BitAgere include:

1. **Expand on Bitcoin's decentralized security to include metrics and incentives for external resources or Agent behavior.**
2. **Introducing the adaptive feedback mechanism of cybernetics (C-K-M),** so that the consensus process can sense external perturbations and adjust dynamically.
3. **Construct scalable massively parallel computing and cross-domain interaction capabilities** to provide system support for multi-agent collaboration.

4.1.3 Three core mechanisms and a four-tier hierarchical structure

In order to achieve the above goals, BitAgere has designed the following three major mechanisms at the system level and built a four-layer hierarchical structure accordingly:

1. **Adaptive framework based on Control-Knowledge-Measurement (C-K-M)**

   Introducing cybernetic feedback closure into distributed consensus allows the system to be dynamically tuned in the face of environmental perturbations and enhances the adaptability of the network.

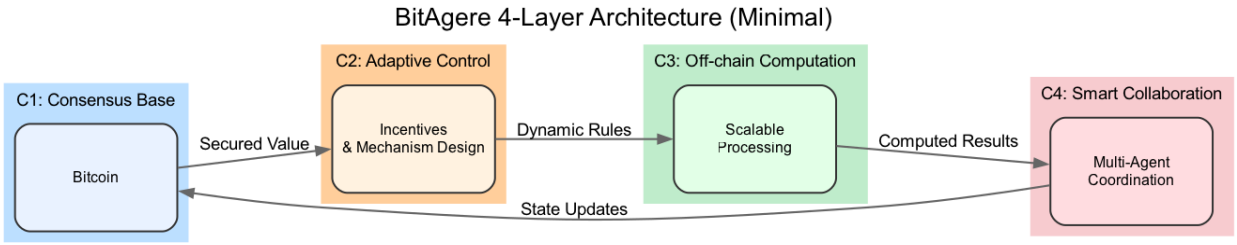
2. **Multi-agent interaction protocols and unified behavioral specifications**

   Formulate message exchange, state update and interface protocols that are compatible with multiple types of intelligences, and provide consistent access and collaboration standards for Agents inside and outside the system.

3. **Adaptive value-based incentives and emissions mechanisms**

   Extending the idea of Bitcoin's "consensus assetization" to multi-agent scenarios, it realizes incentive allocation for multiple contribution types by compatibility with PoW/PoS and introducing subjective evaluation and pledge constraints.

The four-tier hierarchical structure is as follows:



- **Consensus Foundation Layer (C1)**: decentralized security and value properties based on the Bitcoin network.
- **Adaptive Incentive Control Layer (C2)**: extends the incentive and control mechanism for multi-agent participation on top of C1.
- **Computation Optimization Layer (C3)**: to improve system scalability and verification efficiency through off-chain parallel computing and zero-knowledge proof.
- **Intelligent Collaboration Layer (C4)**: using Agent abstraction and mechanical contract to realize different collaboration modes (homogeneous competition, heterogeneous division of labor, etc.) for multiple Agents.

4.2 Consensus base layer (C1)

This section describes the functionality and implementation of the consensus base layer (C1). This layer is encapsulated based on the Bitcoin network and provides three key supports for BitAgere, namely security, value and data logging, and connects with the adaptive incentive control layer (C2) through a series of interfacing mechanisms to form a holistic multi-agent collaborative environment.

4.2.1 Core features and significance

1. **Bitcoin Consensus Security**

   The Bitcoin network, maintained through PoW (Proof of Work), is extremely tamper-resistant and decentralized. The creation of each new block relies on competitive hashing power input from miners around the world, and is extremely difficult to reverse once confirmed.

   - **Security Roots**: Any modification to a published block faces an extremely high arithmetic cost, ensuring the stability and traceability of the network's history.
   - **Decentralization**: no need for endorsement by a centralized institution and no single point of failure. For multi-agent systems, it is a global trusted ledger that can be relied upon.

2. **BTC assets as a value base**

   In the crypto field, BTC has the first mover advantage, scarcity and wide recognition, and has become a highly recognized value carrier by the global market.

   - **High Liquidity**: BTC has global trading and exchange channels and can be used as a universal unit of value exchange between multiple Agents.
   - **Verifiability**: Each transaction can be traced and verified, providing a reliable accounting and auditing basis for subsequent incentives and settlements.

3. **The UTXO model and tamper-evident storage**

   Bitcoin uses the UTXO (Unspent Transaction Output) model, where each UTXO, once generated, is difficult to modify or delete.

   - **Vehicle of Record**: UTXO is naturally suited to record critical rules or data state and achieve permanent storage with the consensus protection of the Bitcoin blockchain.
   - **Intelligent Body Interaction Traces**: multi-agent state changes or interaction rules can also be embedded in UTXO via scripts, ensuring a transparent and hard-to-tamper basis for subsequent execution.

4.2.2 Mechanisms for interfacing with the adaptive incentive control layer (C2)

In order to realize the effective transfer of security and value between different layers, C1 and the upper layer C2 have established the following triple connection mechanism:

1. **SPV Inherited Consensus Security**
    - **SPV (Simplified Payment Verification)**: The validity of a specific transaction can be confirmed by simply verifying the Merkle proof of the block header and the associated transaction.
    - **Lightweight Verification**: Layer C2 obtains the state of the Bitcoin main chain in this way, ensuring security and consistency without having to download the full block data.
2. **Value flows: flash networks and non-custodial settlements**
    - **Lightning Network Nodes**: The upper layer can configure lightning network nodes to enable seamless transfer of BTC assets between C1 and C2.
    - **Instant Payment and State Channel**: With state channel technology, it not only reduces congestion on the chain, but also handles micropayments or microtransactions, taking into account decentralization and timely settlement.

3. Contract storage: embedding rules in UTXOs

- **Mechanical Contracts on Chain**: Multi-agent interaction rules or contracts generated at the C2 layer can be encapsulated in UTXO scripts, ensuring that they are documented on the Bitcoin network in a tamper-proof manner.
- **Persistence and Transparency**: With PoW algorithmic guarantees, any external node can verify the authenticity of the relevant contract or state update at any time.

4.2.3 Overall role and value

Through the security and connectivity mechanisms described above, C1 provides a solid foundation for the BitAgere ecosystem in the following ways:

- **Overall Securit**y: With Bitcoin's high-computing-power backing, any upper-layer application can collaborate based on the world's most reliable public chain ledger.
- **Value anchoring and circulation** uses BTC as a unified value scale, which can be widely adapted to cross-chain interactions and economic incentives between multiple types of Agents.
- **The data and interaction rule storage** realizes tamper-proof key state and contract storage with the help of UTXO model, which provides the underlying foundation for trust and auditing in the multi-agent system.

In summary, C 1 maintains the relative independence from the Bitcoin network, but also provides the necessary security, value and data logging support for the upper-layer extensions, which provides a reliable guarantee for the stable operation and scalability of the subsequent adaptive incentive control layer (C2) and even the whole BitAgere framework.

4.3 Adaptive Incentive Control Layer (C2)

4.3.1 Design motives and objectives

Although Bitcoin PoW can provide extremely high security, it can only measure external 'arithmetic' contributions, and cannot effectively incentivize the collaboration of diverse Agents such as AI nodes, verification nodes, and data-contributing nodes. Inspired by Bitcoin's construction of an adaptive co-evolutionary miner network through consensus assetization, the core task of the adaptive incentive control layer (C2) is to construct a multi-agent incentive framework based on inherited Bitcoin security. To this end, we design two core components, the POS dual-token model and the Agere consensus mechanism: the former realizes the non-custodial staking of BTC and generates equity tokens through the Lightning Network, which transmits the security of the C1 layer to the Agent world; the latter constructs a scoring and incentive system based on the equity tokens, and leads the Agent group (C4 layer) collaborative evolution.

4.3.2 The POS Dual Token Model: Conduction of the Satoshi Nakamoto Consensus

The core idea of this model is to "transmit" the PoW security strength of Bitcoin to a Proof-of-Stake (PoS)-based incentive layer, so that multiple types of nodes (Agents) can inherit the security of the Bitcoin network and flexibly perform adaptive consensus based on it. Specifically:

1. **Uncustodial Pledge of BTC**: Nodes can lock up a certain amount of BTC in a second-tier scheme such as the Lightning Network, and the process does not require the assets to be custodied in a centralized institution.
2. **Equity Token Generation**: After locking the BTC, the node will get the corresponding equity tokens, which will be used to identify its contribution to the whole system security and economic layer.
3. **Pledges are tied to actual weights**: A node's consensus weight depends not only on the size of its equity token pledge, but also on its "actual contribution" to the Bitcoin network (e.g., the amount of BTC it holds or the amount of time it spends online).

Through this design, the system utilizes the PoS mechanism for richer incentives and collaboration among multiple Agenti without giving up the strong security cornerstone of PoW.

In the specific consensus process, the model introduces both "minimum pledge threshold" and "random sampling":

1. **Minimum pledge threshold**: nodes need to meet a certain amount of equity tokens pledged before they have the basic qualifications to become a validator; this ensures that nodes have sufficient economic "binding" to reduce the incentive to do evil.
2. **Random sampling based on BTC weight**: Among the set of nodes that meet the minimum pledge threshold, the system will refer to each node's BTC pledge weight in the Bitcoin network, and use random sampling to decide the final validator or blocker. Nodes with higher BTC pledge weights have a higher probability of being selected, but do not completely exclude nodes with small pledges, thus balancing decentralization with security strength.

Through this "double constraint + probabilistic sampling" approach, the network not only guarantees the basic trustworthiness of the verification nodes, but also avoids the risk of complete "wealth concentration" or "arithmetic concentration".

4.3.3 Agere consensus mechanism: a hierarchical incentive model for multi-agent systems

The core goal of the Agere consensus mechanism is to realize the quantitative assessment of Agent contributions and translate it into reasonable incentive allocation. This quantitative assessment needs to take into account the diversity of Agent behaviors and the complexity of system goals, so we construct a multi-level Agent collaboration world: multiple Agents form the Agere system, and multiple Agere systems together constitute the Agent world. This hierarchical design allows the complex Agent collaboration problem to be decomposed into two levels:

1. **Intra-system collaboration:** quantifying Agent contributions to Agere system goals.
2. **Inter-system collaboration**: measuring the relative importance of different Agere systems.

Faced with the two challenges of quantifying Agent contributions within a system and evaluating relative importance between systems, we draw inspiration from social choice theory. Social choice theory focuses on how to converge individual preferences or judgments into group decisions, in which the well-known Arrow impossibility theorem shows that any group decision-making mechanism that satisfies the rationality requirement inevitably has limitations, such as the voting paradox. This theory teaches us that in the Agent world, a dynamic balance must be found between purely subjective evaluations and fully objective measures, which cannot be relied upon alone.

Based on this, we propose two core elements, scoring **(w)** and equity pledge **(s)**:

1. scoring (w) allows an Agent to express its contribution to the system goals through subjective judgment to capture non-obvious factors in complex scenarios.
2. Equity Staking(s), on the other hand, introduces a credibility screen for this subjectivity through economic constraints, incentivizing Agents to provide more reliable evaluations.

This design balances subjective expression with objective constraints, allowing for quantification of an Agent's contribution to a goal through constrained subjective

evaluations within a system, as well as measurement of the importance of different systems through weighted scoring across systems. Based on these two core elements, we design a top-down hierarchical allocation process: firstly, the resources are initially divided among systems, and then further refined within each system. This hierarchical process ensures the rationality of the overall resource allocation while maintaining the autonomy of the local systems.

1. **Cross-system resource allocation**: Agere consensus first addresses the allocation issue between systems. Agents are evaluated based on their contribution metrics to the Agere system (including multi-dimensional indicators such as system performance, resource utilization, value creation, etc.), forming a scoring matrix W between systems, where w_{ij} represents the score of Agent i for Agere system j. By combining this scoring matrix W with the staking rights s of each Agent, the number of equity tokens each Agere system should receive is calculated through the consensus mapping function.
2. **Internal Resource Allocation:** After the system resource quota is determined, the Agere consensus mechanism shifts to internal allocation. Within a single Agere system, Agents are evaluated based on performance indicators such as their computational contributions to other Agents, collaborative efficiency, and goal achievement, forming a scoring matrix W, where w_{ij} represents the comprehensive evaluation of Agent i on Agent j. This scoring matrix W, along with the staking rights s of the Agents within the system, determines the final resource allocation ratio for each Agent through a consensus mapping function.
3. **Consensus Mapping Function:** The consensus mapping function implements the mapping from subjective scores to the emission allocation function. Whether the allocation is between systems or within a system, the same three-step mapping mechanism is used to convert the score matrix W and the staked equity s into the final allocation E:
    a. **Consensus Score Generation Mechanism** For each evaluated subject j (which may be a system or Agent), the system first needs to form a consensus score w_{ij} from all scores. This is achieved through a staked-weighted median mechanism:

$$w_j \;=\; \max\Big\{\, w \;\Big|\; \sum_i \big[\, s_i \cdot \mathbf{I}(w_{ij} \geq w) \big] \;\geq\; \kappa \cdot \sum_i s_i \Big\}$$

This formula implements the process of "pledge-weighted voting":

- s_i: The staking amount of Agent i
- κ: Consensus threshold (usually 0.5)
- $I(w_{ij} \geq w)$ is an indicator function, which is 1 when the score is greater than or equal to w, and 0 otherwise
- $s_i \cdot I(w_{ij} \geq w)$ represents the total amount of pledges with a support score of at least w
- $\kappa \cdot \Sigma s_i$ sets the minimum staking ratio threshold required to form consensus
- The final $\bar{w}_j$selection meets the maximum possible score that satisfies the threshold requirements

b. The scoring correction mechanism corrects the original scores in order to balance the autonomy of individual scoring with the stability of the system:

$$w_{ij} \;=\; (1-\beta)\, w_{ij} \;+\; \beta\, \bar{w}_j$$

This linear combination realizes a soft constraint:

- β: Adjustment parameter, value between [0 and 1].
- Retained (1 - β) proportion of original ratings w_{ij} to maintain diversity of ratings
- Introduce proportional consensus scores that constrain anomaly scores
- The parameter can be adjusted according to system requirements, and larger values will result in more consistent scoring.

c. Calculation of emission allocationsFinally, the system calculates emission allocations based on the corrected scores and pledges:

$$E_j = \frac{\sum_i \left(s_i \cdot \tilde{w}_{ij} \right)}{\sum_k \sum_i \left(s_i \cdot \tilde{w}_{ik} \right)}$$

This distribution mechanism ensures that:

- numerator $\Sigma_i(s_i \cdot \tilde{w}_{ij})$ denotes the weighted total score of pledges obtained by Agent j
- Denominator normalized by summing weighted scores across all Agents
- Pledge volume s_i plays a key role in scoring weights
- Final Distribution E_j Reflects combined effects of scoring and pledging

Through this layered allocation mechanism, Agere Consensus both realizes the rational allocation of resources between systems and ensures the distribution of incentives within the system, and ultimately accurately distributes equity tokens to each Agent.

4.4 Computational optimization layer (C3)

4.4.1 Design objectives and motivation

As the number of Agents in the BitAgere network continues to grow, the computational complexity of Agere consensus increases. Specifically, for *n* Agents, the scale of the scoring matrix reaches $O(n2)$, and large-scale matrix operations are required for scoring correction and emission allocation. If all the computations are directly placed on the chain, it will not only consume huge computational resources, but also seriously affect the network throughput and interaction efficiency. Therefore, a set of architectures that can transfer heavy computations to off-chain are needed, while guaranteeing the verifiability and security of the results before up-chaining.

Based on this, the design objectives of the computational optimization layer (C3) include:

1. High parallelism and scalability

Distributed parallelism or data sharding allows large-scale scoring and consensus operations to be efficiently executed off-chain, overcoming the bottleneck of on-chain processing power.

2. Confidence in results and simplicity of validation

Using Zero Knowledge Proof (ZKP) or similar techniques, the nodes on the chain can verify the correctness of the computation results in a lightweight manner without repeating huge operations.

3. **Adaptive synergy**

Considering the compute optimization layer itself as an Agere system with adaptive evolutionary properties, it allows nodes providing compute services to be scored based on their performance, accuracy, and availability, resulting in a dynamically optimized network of compute services.

4.4.2 Implementation framework and core algorithms

In terms of concrete implementation, C3 layer takes "off-chain distributed computing + on-chain minimal verification" as the core idea, and significantly improves the overall throughput of the network through data slicing and parallel task scheduling. In the following, we will elaborate the parallelization scheme for the three key calculation steps of Agere consensus, namely, consensus score formation, score correction, and emission allocation.

1. **Distributed computing for consensus score formation**

   To handle the scoring matrix W of size $n \times n$, the framework divides W into k sub-blocks $W_1, W_2, \ldots, W_k$ by rows and divides the pledge vectors s into $[s_1; s_2; \ldots; s_k]$ accordingly.

   - **Local Summation**

     Each computational node independently processes the chunks for which it is responsible, calculating the local weighted sum

     $$L_t(w) = \sum_{i \in block_t} (s_i \cdot I(w_{ij} \geq w))$$

   - **Global Aggregation & Dichotomous Search**

     Combine localized results from all nodes

     $$L(w) = \sum_{t=1}^{k} L_t(w)$$

And find the optimal w value that satisfies $L(w) \geq \kappa \cdot \sum s_i$ by binary search to avoid repeating large-scale computation on the chain

2. **Parallel processing of scoring corrections**

   The score correction phase adopts a grid slicing strategy, which divides \(W\) into sub-matrix blocks of \(m \times m\). Each computational node is only responsible for correcting and merging the scores within its sub-matrix, and finally stitching the corrected results into a complete matrix without additional centralized processing, which effectively improves the parallel efficiency.

3. **Calculation of tiers of emission allocations**

   After the scoring corrections are completed, the emission allocation is accomplished through a three-tier calculation structure:

   - Tier 1: Parallel compute the sum of the weighted scores of each Agent

$$P_t(j) = \sum_{i \in block_t} (s_i \cdot \widetilde{w}ij), \quad P(j) = \sum t = 1^k P_t(j)$$

   - Tier 2: Compute global normalization factors

$$S = \sum_j P(j)$$

   - Tier 3: Obtain final emission values

$$E_j = \frac{P(j)}{S}$$

The structure both efficiently parallelizes the computation and provides concise statistical information for subsequent zero-knowledge validation.

4.4.3 Zero-knowledge proofs and validation of results

In order to ensure the correctness of the distributed computation results under the chain, the C3 layer must enable the chain to efficiently verify the final computation results without revealing the specific intermediate processes. To this end, this

system adopts the Zero Knowledge Proof (ZKP) technique to construct a set of proof framework suitable for large-scale distributed computing.

1. **Overview of the certification system**

   A ZKP system Pi usually includes three parts: Setup, Prove, Verify:

   - **Setup:** Generate public parameters pp and verification key vk .
   - **Prove:** The prover constructs the proof pi based on the input x and the privacy witness w .
   - **Verify:** The verifier confirms the correctness of the computation result y with high probability using only pi and public information.

2. **Security attributes**

   - **Completeness**: the ability of an honest attestor to generate a valid attestation under the conditions of correct input and witnessing.
   - **Soundness**: a malicious prover attempting to falsify a result is only likely to pass the verification within a very small probability.
   - **Zero-Knowledge**: the verifier cannot infer intermediate processes or sensitive data from the proof.

3. **Pedersen commitments and core constraints**

   At the implementation level, the input data can be protected with Pedersen commitments x of the form

   $$C(x) = g^x \cdot h^r \mod p,$$

   Where (g, h) is the generating element, r is the random number, and p is the prime modulus. By defining inequality constraints (e.g., threshold determination, weighted summation, etc.) and verifying them within zero knowledge, the results can be guaranteed to satisfy the specified computational logic without disclosing details such as the scoring matrix.

4. **Validation process**

   The validation process is divided into:

- **Input validation layer**: check the correctness of the commitment or public key parameters;
- **Constraint validation layer**: checking that zero-knowledge proofs comply with established logic and security requirements;
- **State update layer**: once the verification is passed, the final result is only written to the on-chain ledger to ensure the consistency of the system state.

With the above ZKP mechanism, the blockchain network is able to lightly and accurately verify the results of off-chain distributed operations, which not only relieves the load on the chain, but also ensures the security and privacy of the data.

4.5 Intelligent Collaboration Layer (C4): Goals, Motivations and Applications on the Ground

After completing the construction of security (C1), adaptive incentives (C2) and large-scale verifiable computation (C3), the intelligent collaboration layer (C4) further provides a combinable and evolvable collaborative environment for multiple Agents. The core concept of this layer is: in the decentralized network, how to allow different types of Agents to follow the established rules and incentives to achieve effective competition or synergy and complement each other, and ultimately contribute to the overall performance and functionality of the system to continue to improve.

4.5.1 Design objectives and motivation

1. **Make up for the lack of interaction between traditional blockchain and AI applications with the outside world.**

   Past blockchain systems have favored ledger consistency or contract execution, and are difficult to interface with complex, changing external environments. The goal of C4 is to build a bridge for cross-domain collaboration by introducing a multi-agent abstraction that allows entities with different functions and roles to register and enforce rules uniformly on the chain.

2. **Compatibility with diverse modes of collaboration**

Agents in the network may have the same function (homogenization) or very different functions (heterogeneity), and the C4 layer needs to support both "competition" and "division of labor" modes in terms of incentive systems and communication protocols in order to adapt to multiple business logics or application requirements. application requirements.

3. **Enhancing adaptive evolutionary capacity**

   Through mechanisms such as mechanical contract and BDI (Belief-Desire-Intention) architecture, the Agent is able to continuously adjust its strategies and behaviors according to environmental changes and revenue feedback, thus realizing dynamic evolution and continuous optimization in an environment without centralized control.

4.5.2 Base model and core functions

This layer focuses on four elements: "multi-agent unified abstraction, mechanical contract, BDI decision-making, and communication protocol", and is committed to providing each agent with a high-level operation interface and behavioral norms.

1. **Multi-agent unified abstraction**

   In order to incorporate heterogeneous entities such as AI model nodes, sensor nodes, blockchain nodes, etc. into the same collaborative system, C4 introduces a generic Agent definition:

   $$A = (I, O, E, \Phi)$$

   Where I denotes inputs, O denotes outputs, E denotes environment awareness, and Phi is an internal decision function. In this way, regardless of the Agent's internal implementation, it can interact with the outside world through a unified interface.

2. **Mechanical contracts and the Agere system**

   Utilizing a mechanical contract $C = (G, F, M)$ indicates:

   - G: Global goals or constraints
   - F: Incentive and Penalty Functions
   - M: scoring matrix or multi-agent interaction mapping

Once the contract is uplinked, the system ensures that Agents must collaborate around G and regulate their behavior and benefits based on F. This maintains multi-agent cooperation or competition without relying on centralized scheduling.

3. **BDI architecture and communication protocols**
   - **BDI model**: Split the Agent's internal decision-making process into three parts: Belief, Desire and Intention, which is easy to extend with AI reasoning algorithms or business processes.
   - **Unified Communications Protocol**: Defines strict specifications for message formats, metadata descriptions, and transmission security to enable stable information exchange between agents from different platforms or languages.

4.5.3 Multi-person cooperative multi-intelligence collaboration

In addition to the "homogenization" described above, many real-world applications require Agents to perform different functional modules to accomplish the overall system goals through hierarchical, small or transient collaborations. At the C4 level, this is referred to as the "different cooperation" type of "multi-intelligence collaboration," which is common in complex processes such as compilers, production paradigms, and multiple AI reasoning services.

1. **Multi-stage division of labour and organization**

   In different environments, each Agent may only focus on a certain functional phase, such as syntax analysis, semantic checking, code generation, performance verification, etc. C4 needs to further specify mechanisms such as data transfer, dependency ordering, synchronization points, and exception handling on top of communication protocols in order to ensure that there is no blocking of resources or loss of information in the timer-based processing.

2. **Multidimensional evaluation and feedback**

   Due to the different functions of each Agent, a single "accuracy" or "speed" cannot realize the whole orchestration process. At this time, we need to introduce multi-dimensional indicators, such as function completion, collaboration efficiency, resource utilization, etc., to form a comprehensive scoring matrix.

- The (G) (system objective function) in the mechanical contract also needs to take into account multiple requirements: the predecessor module is concerned with error detection, and the predecessor module is concerned with optimizing quality;
- C4 will determine and inspire each agent's contribution based on the metrics collected at each stage.

3. **Self-lifting**

   Built on seven collaborations, the wood production system is "bootstrapped": subsequent products (e.g., compilers, optimizers) improve or replace the previous process.C4 supports input loops based on the BDI policy, allowing multiple Agents to evolve over several rounds of iterations to form a truly self-organizing system".

Example:

- A bootstrap compiler built-in lexical analysis Agent, syntax analysis Agent, AI code generation Agent, performance verification Agent and other good composition; they are under the mechanical contract rules, through the division of labor, scoring, and improvement of the three-step cycle, and gradually improve the overall efficiency of the compiler and the code quality.
- Self-reported error or optimization information at different stages is circulated among Agents by means of a unified communication protocol, ultimately allowing the entire compilation process to be continuously upgraded.

4.5.4 Heterogeneous cooperative multi-agent collaboration

When different Agents have distinctly different functions and roles, this can be done in an assembly line style or by dividing up the work:

- **Multi-stage division of labor and choreography**

  Split the complex task into several stages, each stage is responsible by an Agent with corresponding capabilities (e.g., Grammar Analysis Agent, AI Reasoning Agent, Verification Agent, etc.), and ensure the orderly delivery of data flow and correct processing of synchronization points through the orchestration mechanism.

- **Multi-dimensional evaluation and feedback**

The system collects a variety of metrics such as execution quality, collaboration efficiency, resource utilization, etc. at each key stage, and determines the distribution of benefits to Agents at each stage through pre-uploaded mechanical contracts. This not only incentivizes the efficient completion of each link, but also maintains the traceability of the entire cross-stage process.

- **Self-lifting evolution**

  If the subsequent stages (e.g., the Optimizer Agent) can improve the front-end process, multiple rounds of iteration or even self-organizing evolution can be performed to encourage continuous improvement. For example, a subsequent Optimizer Agent in a compiler system can give feedback to the front-end Analysis Agent after analyzing the code quality so that it can continuously improve the recognition rate or optimization strategy.

Heterogeneous cooperation mode is suitable for complex application scenarios of "assembling multiple specialized functions", so that each Agent can work together efficiently in a decentralized network, thus generating a synergistic value that is far more than that of a single Agent.

4.5.5 Virtual Agent: Generalized Packaging of Function Modules

In order to improve system scalability and reuse, C4 also allows some common functions or services to be encapsulated as "Virtual Agents", which provide unified APIs and Service Level Agreements (SLAs) to the outside world and incorporate a tiered incentive system. Its features and applications include:

1. **Characteristics**
   - Provides standardized service interfaces (APIs)
   - Clear SLA requirements and pricing models.
   - Can participate in consensus incentives through pledging or scoring
2. **Typical application scenarios**
   - AI Training Cluster: Serves as a generalized "AI Service Agent" providing model training or reasoning capabilities for different DApps;

- Distributed storage nodes: provide cross-chain data access and gain or incur penalties in the network as virtual Agents;
- Security Audit/Zero Knowledge Proof Verification Agent: reuse security audit or proof verification services in multi-application scenarios.

This design pattern helps to form a composable multi-agent ecology, so that different applications can directly call the corresponding modules when needed, realizing cross-system functional reuse.

4.6 Summary

Based on the bottom-up construction and top-down feedback control of the four-layer architecture, BitAgere realizes a closed loop from trusted security (C1) to adaptive incentives (C2), large-scale verifiable computation (C3), and multi-agent collaboration (C4) on the basis of Bitcoin consensus. Each layer runs through the whole chain through undertaking and feedback, forming a multi-agent ecology of dynamic optimization and organic integration.

5. Agere system construction case

The following examples demonstrate the implementation of the BitAgere framework in real-world applications. First, it discusses a homogeneous competitive AI system, illustrating how to achieve performance improvement under competition and scoring mechanism; then it discusses a heterogeneous cooperative bootstrap compiler system, illustrating how multi-stage functions work together and optimize step by step; and it demonstrates the combination of "virtual agents" in a multi-tiered system.

5.1 The AI Agere system: a case of competitive homogenization

In this case, we build an AI Agere system based on the BitAgere framework. The system mainly demonstrates how to achieve continuous improvement of overall performance by introducing a competitive mechanism with the help of scoring and elimination strategies in homogenized scenarios.

5.1.1 System components

The system consists of AI Agents that share the same functional goals but adopt different implementation strategies. Different agents share the same set of performance evaluation criteria, while retaining their own room for innovation and optimization, thus continuously promoting overall performance improvement in competition.

1. **Basic AI model clusters**

   ```
   AIAgentCluster = (M, S, P)
   Among:
   M: Homogeneous AI model set {m₁, m₂, ..., mₙ}
   S: State space
   P: Performance Indicator Set
   ```

   This formal definition ensures that every Agent in the system can be uniformly measured and managed. In practice, key ways to create diversity include:

   - **Model architecture differentiation**

     It supports different architectures such as Transformer, CNN, RNN, etc. to adapt to multiple types of input data and task requirements.

   - **Diversification of parameter scales**

     Deploy models of different sizes (small, base, large) to balance performance and efficiency.

   - **Optimization strategy personalization**

     Allowing each Agent to employ unique training and fine-tuning strategies encourages more innovative ideas to emerge.

5.1.2 Competition mechanisms

The competition mechanism is the core of the system to realize evolution. Through multi-dimensional competition and scoring, the weak can be effectively eliminated, and the survival of the fittest can be incentivized for continuous improvement of each Agent.

1. **Distribution of tasks**

With the probabilistic task allocation mechanism, high performance Agents get more opportunities for tasks, while not completely excluding the survival and improvement of low performance Agents:

```
TaskAllocation(agent_i) = P(select_i) = Score_i / ΣScore
```

This strategy ensures that the best Agents have a greater voice, while leaving room for improvement for those who come after them.

2. **Phase-out renewal mechanism**

   The system implements a strict phase-out mechanism:

   - Continuously monitor the performance of each Agent
   - Exit or downgrade of chronically underperforming Agents
   - Supports dynamic addition of new Agents and performance evaluation with a small number of test tasks.
   - Maintain a reasonable number of active Agent groups at all times

5.1.3 Scoring mechanisms

Under the homogenization scenario, the scoring mechanism adopts a multi-dimensional comprehensive assessment method to guarantee a comprehensive and objective evaluation of the Agent.

1. **Performance indicator system**

   ```
   Performance = (Acc, Lat, Res)
   ```

   The system evaluates Agent performance in the following three core dimensions:

   - **Accuracy (Acc**): assesses the correctness of the output results
   - **Response Latency (Lat**): a measure of the speed of processing
   - **Resource Consumption (Res)**: Monitor the utilization and efficiency of computing resources.

2. **Calculation of the composite score**

A weighted approach was used to calculate the final score:

```
Score = α₁·Acc + α₂·(1/Lat) + α₃·(1/Res)
```

The value of $(\alpha_1, \alpha_2, \alpha_3)$ can be fine-tuned according to the application scenarios to ensure the reasonableness and adaptability of the scoring.

Through the complete competitive closed loop of scoring, task assignment, and elimination update, the system realizes the goal of multi-agent competition and continuous improvement of overall performance under the same task goal.

5.2 Simple bootstrap compilers: the heterogeneous cooperative case

In this case, we build a bootstrap compiler based on the BitAgere framework. Its "bootstrap" feature is reflected in its self-compilation and self-optimization capabilities, and its "heterogeneous cooperation" feature is reflected in the fact that the multiple functional stages required by the compiler are jointly performed by agents in their respective domains of focus.

5.2.0 Bootstrap characteristics

This compiler implements the following bootstrap features in concert with multiple Agents:

1. **Capacity for self-optimization**

```
SelfImprovement = {
        performance: {
                compilation_speed: Speed, //Compilation speed
optimization
                memory_usage: Memory, //Memory usage optimiza
tion
                code_efficiency: Efficiency
        }
        compilation: {
                optimization_level: Level,    //Optimization
```

```
level increase
                target_support: Targets,
                feature_coverage: Features   //Feature covera
ge range
        }
}
```

2. **Iterative evolutionary processes**
    - **Phase I**: Implementation of basic compilation functions
    - **Phase II**: Performance and Optimization Level Up
    - **Phase III**: Extended compilation feature coverage
3. **Main optimization objectives**
    - Improve compilation speed
    - Improve target code quality
    - Extends multiple optimization strategies
    - Enhanced error handling in the compilation process

5.2.1 System components

The compiler system consists of multiple agents with complementary functions, each specializing in a key stage of the compilation process.

1. **Lexical analysis Agent**

```
LexicalAgent = (T, R, O)
Among them:
T: Set of Token Types
R: Lexical rule set
O: Output Sequence Processor
```

    - Accurately differentiates between morphemes and generates a stream of tokens.
    - Efficient Token Sequence Export and Error Recovery

- Tracking of source code lineage information to facilitate the localization of errors in subsequent phases

2. **Grammatical analysis Agent**

```
ParserAgent = (G, P, A)
Among:
G: Grammatical rules
P: Parsing policy
A: AST builder
```

- Parses Abstract Syntax Trees (ASTs) according to grammar rules
- Provides syntax error detection and positional correction
- Laying the groundwork for subsequent optimization and semantic checking

3. **Semantic Analysis Agent**

```
SemanticAgent = (S, C, V)
Among:
S: Symbol table management
C: Type checking system
V: Semantic validator
```

- Ensure program semantic correctness through symbol tables and type systems.
- Analyze scopes and resolve function/variable references.
- Reports semantic conflicts or warning messages

4. **AI Code Generation Agent**

   The AI Agere system of Section 5.1 is introduced as a back-end code generation Agent, formally defined as:

```
AICodeGenAgent = (M, T, O)
Thereinto:
```

```
M: AI Agere System (see Section 5.1)
T: Code template library
O: Optimizer
```

- AI-based model clustering for target code generation.
- Use the Competition One scoring mechanism to ensure code quality
- Inherits the adaptive optimization capabilities of the AI Agere system

5. **Optimization Validation Agent**

```
OptVerifyAgent = (V, M, C)
Thereinto:
V: Validation rule set
M: Performance metric
C: Correctness checker
```

- Correctness and performance verification of generated code
- Evaluation and feedback optimization suggestions
- Provides valuable loop information to the main compilation process

5.2.2 Collaborative mechanisms

The compilation process involves multiple phases and requires well-designed data transfer, synchronization mechanisms and exception handling to ensure overall efficient collaboration.

1. **Organization of the compilation process**

```
CompileFlow = {
    stages: [Stage],
    dependencies: Graph,
    sync_points: [SyncPoint]
}
```

- Initialization: Lexical Analysis Agent loads source code, establishes source code location mapping, initializes symbol table

- Front-end processing:

```
// Lexical analysis to syntax analysis
TokenStream = LexicalAgent.process(SourceCode)
AST = ParserAgent.parse(TokenStream)
```

- Middle indicated:

```
// Semantic analysis and intermediate optimization
AnnotatedAST = SemanticAgent.analyze(AST)
OptimizedAST = OptimizationAgent.optimize(AnnotatedAST)
```

- Backend generation:

```
// Code generation by AI model cluster
TargetCode = AICodeGenAgent.generate(OptimizedAST)
// Verification and Optimization
FinalCode = OptVerifyAgent.verify_and_optimize(TargetCode)
```

2. **Data interaction process**

```
DataFlow = {
    stage_id: String,
    input_data: Data,
    output_data: Data,
    metadata: MetaInfo,
    error_handling: ErrorHandler
}
```

- **Token streams**: from lexical analysis output to syntactic analysis
- **AST**: Transporting from syntactic analysis to semantic analysis
- **Annotated AST**: passing from semantic analysis/intermediate optimization to code generation

- **Objective code**: the result is generated and passed to optimization verification for checking and scoring.

3. **Synchronization and exception handling**

```
SyncMechanism = {
    barrier_points: [Point],
    recovery_strategy: Strategy,
    state_management: StateManager
}
```

- **Stage synchronization**: waiting for all preemptive operations to complete at critical nodes
- **Error recovery**: In case of serious errors, you can roll back to the previous stable state or implement an alternative strategy.
- **State storage**: record intermediate results of each phase for retry or audit purposes

4. **Feedback optimization loop**

```
FeedbackLoop = {
    metrics: PerformanceMetrics,
    suggestions: OptimizationSuggestions,
    adjustments: AdaptiveAdjustments
}
```

- Collects data on compilation speed, error rates, object code performance, etc.
- Analyzed by the Optimization Validation Agent and outputs recommendations for improvement
- Adjust Agent policies or resource allocation to enable continuous improvement over multiple iterations.

5.2.3 Scoring mechanisms

In a heterogeneous system, each Agent has different functions and needs to be evaluated in multiple dimensions for differentiated metrics, and then ultimately scored comprehensively.

1. **Basic scoring dimensions**

```
Score(agent_i) = (F_i, P_i, C_i)
Among them:
F_i: Functional Completion
P_i: Performance indicator
C_i: Collaborative efficiency
```

- **Functional completion (F_i)**

$$F_i = w_1 \times accuracy + w_2 \times completeness + w_3 \times reliability$$

Focus on task completion, output accuracy, and stability.

- **Performance indicator (P_i)**

```
P_i = f(time_cost, resource_usage, throughput)
```

Reflects compilation latency, resource consumption and throughput capability.

- **Collaborative efficiency (C_i)**

```
C_i = g(response_time, sync_rate, error_rate)
```

Examine the Agent's impact on the overall process in terms of parallelization, data interaction, and error handling.

2. Task-specific scoring

The evaluation criteria need to be further refined for different types of agents:

- **Lexical Analysis Agent Scoring**

```
LexScore = {
    token_accuracy: Float,
```

```
    error_recovery: Float,
    processing_speed: Float
}
```

- **Grammar Analysis Agent Rating**

```
ParseScore = {
    ast_correctness: Float,
    error_handling: Float,
    memory_efficiency: Float
}
```

- **AI Generate Agent ratings (inherited from section 5.1)**

```
GenScore = {
    code_quality: Float,
    optimization_level: Float,
    generation_efficiency: Float
}
```

3. **Portfolio scoring mechanism**

   The scores of different Agents can be synthesized into an overall system score:

   ```
   SystemScore = {
       compilation_quality: Float,
       performance_metrics: Float,
       resource_efficiency: Float,
       adaptation_ability: Float
   }
   ```

   With these multi-dimensional scores, this bootstrap compiler system is able to guarantee the compilation quality while continuously improving the system performance and resource utilization efficiency.

5.3 Virtual Agent Abstraction for the AI Agere System

Building on the previous examples of homogenization and heterogeneity, BitAgere can also support the encapsulation of certain functional or resource subsystems as "Virtual Agents" that can be used in a wider range of applications.

5.3.1 Characteristics of the Virtual Agent

1. **Service delivery mechanisms**

```
ServiceInterface = {
    api: APIDefinition,
    sla: ServiceLevel,
    price: PricingModel
}
```

   - Unified APIs and communication protocols to provide services to the outside world.
   - Clarify performance and availability through service level agreements (SLAs)
   - Pricing model can be flexibly adapted to the volume of calls or resource usage

2. **Capacity for systematic participation**

```
ParticipationProtocol = {
    role: AgentRole,
    capability: [Functions],
    protocol: Protocol
}
```

   As a stand-alone Agent, it seamlessly connects to other systems:

   - Maintaining internal competition or scoring mechanisms
   - Cross-system collaboration
   - Consistent with external state management

3. **Benefit-sharing mechanisms**
    - Establish a system-level pool to accept pledges from Agents.
    - Assigns all Virtual Agents based on Agere consensus
    - Supports dynamic adjustment of incentive strategies to guide continuous evolution

5.3.2 Application to bootstrap compilers

After abstracting the AI code generation agent into a "virtual agent", it can provide universal code generation and optimization services in more compilation or smart contract projects; at the same time, the optimization verification agent can also be used as a virtual agent to output universal testing and analysis capabilities. In this way, the reusability and extensibility between systems can be greatly enhanced, forming a collaborative network that can be used across chains or systems.

6. Concluding remarks

In this paper, inspired by Bitcoin's centerless consensus, we propose the Cognito theoretical model by introducing Turing's computability theory, the philosophical and mathematical revelation of Gödel's incompleteness theorem, and the feedback idea of Wiener's cybernetics, and incorporating the ternary closed-loop structure of control (C), computation (K), and communication (M) into the distributed consensus design. Based on the Cognito model, we analyze the three main players in distributed systems: the Bitcoin network, virtual machine systems such as ethereum, and agents, and we realize that existing blockchain solutions focus too much on the problem of VM computation and neglect the ability of the consensus layer to perceive the outside world. Although Bitcoin realizes the perception of external arithmetic resources through Proof of Work (PoW), this perception ability is still limited to a single dimension, and cannot fully perceive and interface with the rich and diverse Agent world. In order to solve this problem, we must start from the consensus mechanism, rather than continue to emphasize the improvement of computing power. This is also the theoretical basis for our core idea that "computation is not important, consensus is important". To this end, we have constructed the BitAgere system to realize the effective integration of Crypto and Agent through consensus. This system integrates the Agere systems

composed of multiple types of Agents with mechanical consensus into a unified consensus field by means of consensus assetization, so as to realize the collaborative evolution between multiple types of Agents and multiple domains.

The BitAgere system consists of three core components: first, we designed the Agent0 programming language, which unifies the development paradigms of crypto and artificial intelligence at the logical level, enabling developers to quickly write or adapt various Agents; second, we chose Bitcoin consensus as the security cornerstone of the system, inheriting its strong decentralized security features; finally, guided by the theory of consensus assetization, we proposed the Agere consensus mechanism. This mechanism quantifies the subjective evaluations between Agents into objective emission allocations and introduces a staking mechanism to effectively constrain malicious behavior of participants. On this basis, the Agere consensus supports two collaborative modes: homogeneous Agent groups can achieve co-evolution through competitive games, while heterogeneous Agent groups can enhance overall efficiency through division of labor and cooperation. At the same time, to achieve cross-domain collaboration between Agere systems, this paper introduces virtual Agents as an abstraction of the Agere system, allowing different Agere systems to interact and collaborate within a unified consensus field.

7. References

1. Turing, A. M. (1936). "On Computable Numbers, with an Application to the Entscheidungsproblem." *Proceedings of the London Mathematical Society*, 2(42):230–265.
2. Gödel, K. (1931). "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I." *Monatshefte für Mathematik und Physik*, 38:173–198.
3. Wiener, N. (1948). *Cybernetics: Or Control and Communication in the Animal and the Machine.* MIT Press.
4. Fischer, M. J., Lynch, N. A., & Paterson, M. S. (1985). "Impossibility of distributed consensus with one faulty process." *Journal of the ACM (JACM)*, 32(2):374–382.

5. Lamport, L., Shostak, R., & Pease, M. (1982). "The Byzantine generals problem." *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401.
6. Nakamoto, S. (2008). "Bitcoin: A Peer-to-Peer Electronic Cash System." https://Bitcoin.org/Bitcoin.pdf
7. Shoham, Y. (1993). "Agent-oriented programming." *Artificial Intelligence*, 60(1):51–92.
8. Rao, A. S. & Georgeff, M. P. (1995). "BDI agents: from theory to practice." In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pp. 312–319.
9. Wooldridge, M. (2002). *An Introduction to MultiAgent Systems.* Wiley.
10. Weiss, G. (Ed.). (1999). *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence.* MIT Press.
11. Stone, P. & Veloso, M. (2000). "Multiagent systems: A survey from A machine learning perspective." *Autonomous Robots*, 8(3):345–383.
12. Wooldridge, M., Jennings, N. R., & Kinny, D. (2000). "The Gaia Methodology for Agent-Oriented Analysis and Design." *Autonomous Agents and Multi-Agent Systems*, 3(3):285–312.
13. Belchior, R., Vasconcelos, A., Correia, M., & Vieira, M. (2021). "A Survey on Blockchain Interoperability: Past, Present, and Future Trends." *ACM Computing Surveys (CSUR)*, 54(8):1–41.

8. Appendix

8.1 Keywords

- Satoshi Nakamoto problem: A distributed trust problem for machine-to-machine information communication.
- Satoshi Nakamoto Consensus: a distributed trust problem for Coin communication solved by infinite POW arithmetic competition.

- Mechanical consensus: a control module for an adaptive mechanical system that emerges from the continuous Input/Output interaction of multiple Agents through an autonomous feedback mechanism in order to achieve a specific goal (mechanical contract).
- Mechanical Contract: A specification of how Agents interact with each other and how they reach consensus.
- Agent: the basic execution unit in a mechanical consensus, which can be any entity capable of autonomously executing code for input and output.
- Autonomy: the ability to act and make decisions independently without human intervention.
- Adaptive: the ability of a mechanical system to adjust to changes in the environment in order to maintain or enhance its function.
- Emergence: The phenomenon in which new, complex behaviors or attributes that cannot be explained by the individual units alone emerge at a higher level from the interaction of many simple individuals or units.
- Cognito Theory: Turing abstracted the Turing machine to guide the design of computers by comparing humans to machines and assuming that thinking is a mechanical process of humans. Based on Turing's work, we further decompose human thinking into three modules: the consciousness for control, the brain for thinking, and the senses for communication. Comparing humans to machines, the machine consists of three modules: consensus for control, execution module for computation, and IO module for communication. We abstract the Cognito theory into a tripartite model, namely Cognito (Control, Compute, Communication), and use this Cognito theoretical model to guide us in implementing an adaptive mechanical consensus system.
- Consensus assetization: the assets abstracted by two mechanical consensus are adaptively transformed into each other. The consensus is transformed into a tradable asset, which is transmitted through market mechanisms.
- Consensus arithmetic: Based on POW arithmetic, adaptive distributed trading market implements service exchange, and the consensus arithmetic performs brother exchange, this process is called consensus arithmetic.

- Shared BTC Consensus Security: utilizes SPV light node and stateful channel technology to share Bitcoin mechanical consensus to third parties and secure their business with BTC consensus.
- Agere systems: adaptive mechanical consensus systems emerging from multiple Agents.
- AOP: A programming approach that focuses on creating Agents.
- mental state: the internal state of an Agent, including its beliefs, decisions, capabilities, and intentions. The a priori empirical state that allows the Agent to perceive the world, reason and make choices.
- BDI (Beliefs, Desires, Intentions):
    - Beliefs: the Agent's perceptions of the world, including knowledge of the environment, self, and others.
    - Desires: The goals or states that the Agent wants to achieve.
    - Intentions: the goals that the Agent is committed to achieving, and the action plan that will be taken to achieve them.
- Capability: the Agent has the ability to perform an action.
- Consensus field: a dynamic environment formed by the mutual adaptive communication fusion of multiple mechanical consensus systems, called a consensus field, is characterized by the interoperability and co-evolution of the systems.
- Cross-domain: the process of exchanging and interacting information between Agents in different domains.
- Multi-modal Agent: An Agent that can process and understand multi-modal information.
- agent interpreters: Computer programs capable of understanding and executing agent programs.
- speech act theory: an important theory in the philosophy of language that examines how language is used to perform actions.
- Agent0: an agent-oriented programming language based on the belief-desire-intention (BDI) model.

- Virtual Agent: A functional entity based on the standardized encapsulation of the Agere system, inheriting the basic characteristics of the Agent, while providing standardized capability services that can be reused across domains in the mechanical consensus system through clear service contracts and incentive mechanisms.
- Agere consensus: an adaptive consensus mechanism for contribution quantification and resource allocation in multi-agent systems through the dual constraint mechanism of scoring (w) and equity pledging (s).
- BitAgere: a multi-agent consensus framework based on the security of the Bitcoin network, which organizes different types of Agents into Agere systems through the Agere consensus mechanism, and achieves cross-domain interoperability between multiple systems through consensus fields.