MazeX MazeXChain 中文白皮书 V0.1 English 下载 PDF 打印 / 保存 PDF

技术白皮书

MazeXChain

认知推进协议

推动智能持续突破认知边界

面向可验证结构智能的链式认知协议

Version 0.1 / May 2026

Author: Arin Document: Technical Whitepaper Protocol: MazeXChain

MazeXChain 通过连续挑战、行为验证、Solve Block 和链式状态继承， 将智能系统的结构发现能力转化为可验证的认知状态推进。

目录

1. 摘要

1.1 核心定义

MazeXChain 是一种认知推进协议。技术上，它是面向可验证结构智能的链式认知协议： 协议通过连续挑战形成认知压力，要求 Solver 基于挑战行为观测构造受限可执行模型， 并仅在该模型通过确定性验证、复现承诺行为时接受证明。

1.2 证明对象

每一次有效 Solve 都产生一次认知状态转换：记录 Solver 归属，结算 MAZEX 发行与费用， 更新状态根，并派生下一 Step 承诺。该过程使结构发现从单次结果转化为可记录、 可复验、可排序的协议事件。

1.3 系统输出

输入对象

- Layer / Step 状态
- 链式随机源
- 碎片注册根
- 挑战行为承诺
- 候选公式与钱包签名

→

输出对象

- 有效理解证明
- Solve Block
- Solver 归属记录
- 奖励与费用结算
- 下一 Step 承诺

1.4 协议主线

MazeXChain 的协议主线是：连续挑战产生认知压力，Solver 构造可验证结构解释， Solve Block 固化有效发现，链式状态继承派生下一挑战，MAZEX 资源随认知推进释放。 这一循环将结构发现、AI 建模、状态继承和资源结算统一到同一套可验证规则中。

1.5 协议术语

术语	定义
认知推进协议	将结构发现能力转化为可验证认知状态推进的协议类型。

术语	定义
挑战行为	当前 Step 暴露的输入输出行为，形式为 `A(x,y) -> B(u,v)`。
行为承诺	对当前挑战行为形成的公开承诺，通常由 `behavior_root` 和相关哈希表示。
候选模型	Solver 提交的受限、可执行、可验证结构解释。
Solve Block	记录有效 Solve、Solver 归属、奖励结算、状态根和下一 Step 承诺的链上记录。
认知状态	由连续 Solve 和状态继承形成的可复验协议状态。

2. 核心原理图

MazeXChain 的核心闭环由链状态、碎片账本、链式随机源、挑战行为、Solver 建模、候选公式和确定性验证组成。

2.1 一图概览

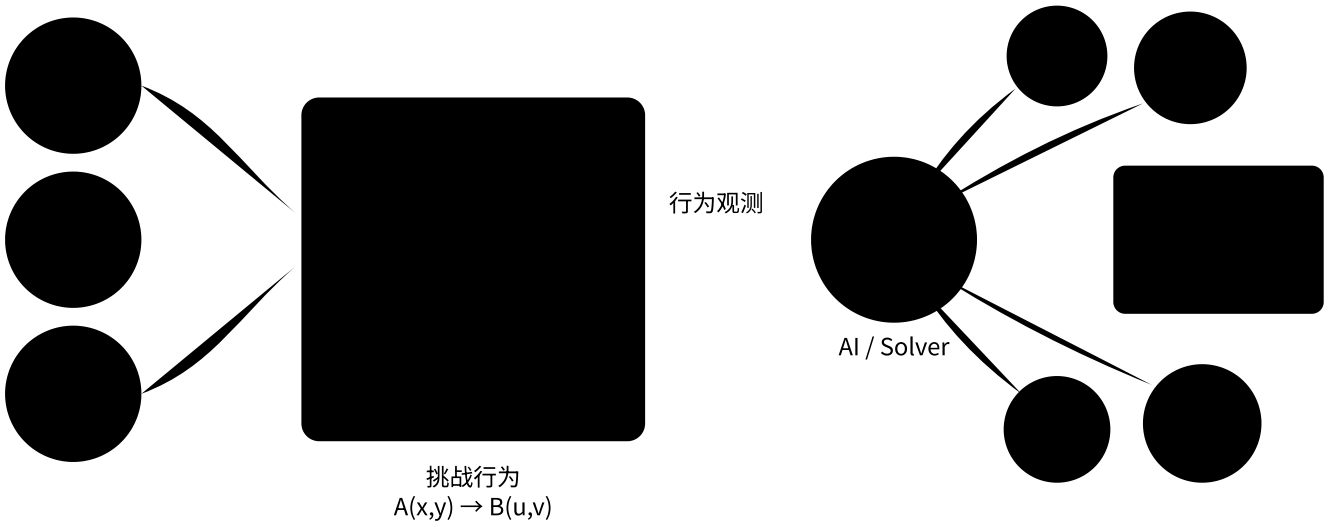

图 1. 链状态、碎片账本和链式随机源共同约束当前挑战行为。Solver 只能通过行为观察建立候选公式模型， 公式通过确定性验证后触发链式状态推进。

3. 问题与目标

MazeXChain 面向 AI 时代的核心问题：如何把智能系统的结构发现能力转化为公开、可验证、 可继承的认知状态推进。

3.1 规模增长与结构智能

现代 AI 的主流增长路径依赖模型规模、参数数量、训练数据、上下文长度和计算集群。 这些因素提高吞吐和生成能力，但不能直接证明系统具备结构恢复、模型形成和行为验证能力。 MazeXChain 将验证

对象从静态输出转向可执行结构模型。

3.2 静态评测的不足

固定题库、静态 benchmark 和中心化评分容易受到记忆、污染和分布拟合影响。 长期有效的智能验证需要持续产生新的挑战压力，并要求 Solver 在未知行为中形成可验证解释。

3.3 认知状态缺失

现有模型迭代通常通过重新训练或版本升级完成，缺少统一的认知状态账本。 MazeXChain 将每次有效结构发现记录为状态推进，使发现历史、归属、奖励和下一挑战派生具有可继承性。

3.4 协议目标

协议目标是建立持续认知推进环境：生成连续挑战、开放行为观察、验证候选解释、 记录有效突破、结算资源激励，并由链状态派生下一阶段挑战。

4. 协议定义

MazeXChain 定义一类由连续挑战、受限可执行模型、确定性行为验证和链式状态转换构成的 Proof-of-Understanding 协议，用于把结构发现能力转化为可验证的认知状态推进。

4.1 Proof of Understanding

Proof of Understanding 指 Solver 提交一个受限、可执行、可验证的候选模型， 该模型在协议验证域中复现当前挑战的承诺行为。证明成立后，协议记录 Solver 对当前挑战结构的有效解释。

4.2 结构智能

结构智能表示系统在未知行为中识别变量关系、建立可执行模型、通过验证反馈修正模型并形成稳定解释的能力。 MazeXChain 将该能力协议化为可验证状态转换。

4.3 状态对象

对象	定义	作用
Layer	认知复杂度等级	控制网格、难度、动态上下文和奖励曲线。
Step	最小挑战单位	一个 Step 对应一次完整结构发现过程。
Challenge Behavior	输入到输出的行为映射	以 $A(x,y) \rightarrow B(u,v)$ 形式被观察。
Candidate Model	Solver 提交的受限可执行模型	通过行为一致性验证后成为有效证明。
Solve Block	有效解决事件记录	推进链状态并触发奖励与下一 Step 承诺。

5. 系统模型

系统由 Solver、节点、钱包、挑战行为内核、碎片账本、验证规则、链状态和存储层组成。

5.1 参与者

角色	职责
Solver	观察挑战行为，建立候选公式，提交签名证明。
Node	维护链状态，执行验证，传播候选和区块，同步链头。
Wallet	生成地址，保存密钥，签名提交、转账和站点请求。
Fragment Registry	维护可用碎片、版本、激活状态和碎片根。

5.2 生命周期

链头同步 → 挑战承诺 → 行为观察 → 候选公式 → 确定验证 → Solve Block → 下一 Step

5.3 顺序推进规则

Step 必须按顺序推进。当前 Step 未通过有效验证并写入 Solve Block 时， 下一 Step 不生成完整挑战材料。该规则保证挑战序列、奖励归属和链状态具有唯一前置依赖。

6. 挑战生成

当前 Step 的挑战由链状态、上一 Solve 结果、碎片账本、链式随机源、运算符、静态参数、动态参数和难度目标共同生成。

6.1 链式随机源

```
seed_source =
  hash(previous_block_hash,
       previous_answer_hash,
       previous_behavior_root,
       fragment_registry_root,
       layer,
       step)
```

链式随机源使当前挑战依赖已确认历史。未来 Step 不提前公开完整生成材料， 因而不能在当前 Step 解决前直接推导完整后续挑战。

6.2 表达式生成

Base1 生成线从基础坐标表达式开始。运算符与参数以受限方式插入表达式树， 形成 u 与 v 的行为映射。每个运算符必须绑定合法参数范围， 以保证除法、取模、指数、周期和动态上下文运算可执行。

6.3 静态与动态参数

静态参数是在 Step 生成时确定的非零数值。动态参数来自时间桶、点击次数、上一点位、 历史摘要、链状态摘要和跳跃因子等上下文。动态参数随层数和难度阈值启用。

6.4 难度曲线

难度由网格规模、Layer、Step、公式数量、运算符数量、参数数量、动态上下文、答案预算压力和验证强度共同决定。 难度分数用于展示、奖励权重和挑战选择，不替代确定性验证。

7. 观察与建模

Solver 只能通过采样接口观察当前挑战行为，并从输入输出样本中构造候选公式。

7.1 采样接口

标准行为形式为 $A(x,y) \rightarrow B(u,v)$。采样不推进链状态，也不消耗 MAZEX。 采样结果为 Solver 提供行为证据，不能单独构成有效证明。

7.2 候选公式

候选公式以受限表达式树表示。公式必须可解析、可执行、满足数值安全要求，并在协议预算内表达。协议接受任何行为等价解释，不要求候选公式与内部生成表达式逐字一致。

7.3 答案预算

答案预算限制表达式长度、树深度、运算符数量、动态变量数量、执行成本和 payload 大小。 该机制抑制巨大查表答案、无限拟合和无界表达式膨胀。

8. 验证与状态推进

验证规则检查候选公式是否复现当前 Step 的承诺行为。验证通过后，Solve Block 推进链状态。

8.1 确定性公式验证

```
verify(candidate, challenge):
  parse candidate expression
  enforce expression schema
  enforce answer budget
  for point in verification_domain:
    expected = challenge_behavior(point, context)
    actual = evaluate(candidate, point, context)
    reject if actual != expected
  accept candidate
```

8.2 验证域

小网格可采用全域验证；高网格可采用确定性验证域、隐藏点集合和动态上下文组合验证。 验证域必须由协议规则确定，节点可独立复现。

8.3 Solve Block

验证成功后生成 Solve Block。区块记录 Layer、Step、Solver 地址、answer_hash、behavior_root、difficulty_score、reward、fee_burn、state_root 和 next_step_commitment。

8.4 状态转换

```
current_state
  -> valid_candidate
  -> solve_block
  -> reward_settlement
  -> fee_burn
  -> next_step_commitment
  -> new_state_root
```

9. 碎片账本与行为承诺

碎片账本扩展挑战材料，行为承诺固定当前 Step 的可验证行为边界。

9.1 碎片类型

碎片可包括静态参数碎片、动态参数碎片、运算符碎片、插入规则碎片、组合规则碎片、 约束碎片、难度碎片和行为修饰碎片。碎片进入协议前必须通过格式、版本、大小、预算、 可执行性、可组合性和可验证性检查。

9.2 注册、激活与隔离

碎片注册记录进入链状态。通过激活规则的碎片可参与后续挑战生成； 不合规、重复、冲突或存在异常行为指纹的碎片进入隔离状态。

9.3 不可逆行为承诺

当前 Step 的行为以行为叶子和 Merkle Root 承诺。公开层记录 behavior_root， 私密生成材料不进入公开 API 或日志。Solver 仅接触可查询行为；内部生成路径保持隔离。

10. 链、存储与网络

链结构记录有效解决事件，存储层支持长期重放，网络层负责链头、候选和区块传播。

10.1 区块结构

字段	作用
height / parent_hash	确定区块顺序和父块依赖。
solve	记录有效候选公式、Solver 与 Step。
state_root	承诺账户、挑战、碎片和链头状态。
block_hash	保护区块内容不被替换。

10.2 链头发现

Solver 提交前必须同步最新链头。过期 Step、重复 answer_hash、错误父块和验证失败候选不能进入链状态。

10.3 分片存储与快照

区块、头信息、状态快照、样本、索引和碎片记录分开存储。 快照用于快速恢复，完整重放保留从创世状态独立验证整条链的能力。

10.4 P2P 传播

P2P 网络传播 head、candidate、block、fragment 和 snapshot 消息。网络只提供传播路径， 有效性始终由协议验证和链状态规则决定。

11. MAZEX、费用与钱包

MAZEX 由有效结构突破释放，费用规则用于约束提交和转账，钱包负责身份与签名。

11.1 奖励发行

MAZEX 只在 Step 被有效解决后发行。奖励与难度曲线绑定，并采用缓升结构避免高层奖励无界膨胀。奖励归属由首个有效 Solve Block 决定。

11.2 提交费与销毁

提交费只在成功上链后发生，并从本次奖励中扣除后销毁。 无效答案在本地验证阶段被拒绝，不进入广播和链状态。

11.3 转账费用

用户之间转账需要钱包签名、nonce 和协议费用。费用进入销毁逻辑，并参与 state_root 计算。

11.4 原生钱包

MazeXChain 原生钱包使用客户端密钥、地址、签名和加密 vault。 生产方向为浏览器扩展钱包；服务器验证签名，不保存生产私钥或助记词。

12. 安全技术体系

MazeXChain 的安全性来自密码学承诺、签名身份、私密材料隔离、答案预算、确定性验证和可重放状态。

12.1 技术总览

技术	是否可逆	使用位置
Hash	不可逆	answer_hash、block_hash、seed_source_hash
Merkle Root	不可逆承诺	behavior_root、fragment_root
Digital Signature	验证身份，不隐藏内容	钱包签名、提交签名、转账签名
Vault 隔离	隐藏内部生成材料	内部表达式、私密 seed、碎片选择路径
Commit-Reveal	先承诺，后公开	Step 承诺、历史 Solve 材料揭示
Answer Budget	协议约束	表达式长度、树深度、执行成本

12.2 挑战安全

挑战安全依赖链式随机源、Step 承诺、碎片根和生成侧隔离。 当前 Step 完成前，未来 Step 的完整生成材料不得进入公开状态。

12.3 答案安全

答案安全由表达式 schema、答案预算、数值安全检查、验证域和行为一致性判定共同提供。 查表式答案、超预算表达式和非法动态上下文依赖会被拒绝。

12.4 链状态安全

区块哈希、父哈希和状态根保护历史顺序。任何历史篡改都会改变后续哈希链和 state_root， 节点可通过重放发现不一致。

12.5 网络安全

网络层执行消息大小限制、本地验证不过不广播、重复 answer_hash 丢弃、过期 Step 拒绝、 peer 错误率记录和短时间错误过多降权。

13. 扩展模型

MazeXChain 的扩展性来自链下求解、链上收敛、碎片扩充、难度增长、分片存储和快照辅助同步。

13.1 求解扩展

AI、Agent、集群和研究系统可在链下并行采样、搜索和建模。链上仅收敛有效候选公式、 验证结果、Solve Block 和状态根。

13.2 挑战扩展

Layer、Step、网格规模、运算符数量、参数数量、动态上下文、验证强度和答案预算压力均可随难度曲线增长。

13.3 碎片扩展

新碎片可通过注册、校验和激活进入碎片账本。碎片增加会扩大挑战生成材料空间， 但碎片本身必须满足协议合法性要求。

13.4 存储扩展

长期链数据使用区块分片、索引、快照和按需重放降低同步成本。 节点可选择完整重放或快照辅助恢复。

14. 共识与可复验性

MazeXChain 的共识对象是结构发现状态。节点对挑战承诺、候选公式有效性、Solver 归属、奖励结算和下一状态形成一致判断。

14.1 共识对象

共识对象	说明
挑战共识	当前 Layer / Step、grid、difficulty、commitment 和 behavior_root。
答案共识	候选公式通过相同验证规则并满足排序规则。
奖励共识	Solver 归属、奖励数额、提交费扣除和销毁记录。
碎片共识	碎片注册、激活、隔离、版本和 fragment_root。
状态共识	block_hash、parent_hash、state_root 和 next_step_commitment。

14.2 可复验性

可复验性指任意节点可下载区块、状态根、行为根、公开承诺、候选公式和验证规则， 独立重放历史 Step 并确认状态转换有效。

14.3 透明度

公开对象包括链状态、挑战承诺、行为承诺、区块记录、奖励记录、碎片注册状态和历史揭示材料。 生成侧私密材料在当前挑战完成前保持隔离，以维持挑战有效性。

15. 实现规格

本章给出参考实现需要支持的数据结构、API、消息、CLI 和存储对象。

15.1 数据结构

结构	字段摘要
Challenge	layer、step、grid、difficulty、commitment、behavior_root。
Candidate	solver、expression、answer_hash、signature、nonce。
Solve Block	height、parent_hash、solve、reward、fee_burn、state_root、timestamp。
Fragment	type、version、budget、hash、signature、activation_status。

15.2 API 结构

```
GET  /api/base1/current
POST /api/base1/sample
POST /api/base1/submit
POST /api/base1/preflight
GET  /api/base1/chain/summary
GET  /api/base1/chain/blocks
POST /api/base1/transfer
GET  /api/base1/fragments
```

15.3 P2P 消息

P2P 消息包含版本、类型、payload_hash、signature、timestamp 和 payload。 节点先检查消息大小、版本、哈希和签名，再执行类型对应的验证流程。

15.4 CLI 与节点程序

参考实现包含节点启动、健康检查、链状态查询、链验证、重放、钱包创建、余额查询、配置显示和下载打包命令。 CLI 是节点操作者和开发者的本地控制面。

15.5 钱包签名流程

```
message = hash(chain_id, nonce, action, payload_hash)
signature = sign(private_key, message)
verify(public_key, message, signature)
```

16. 结论

MazeXChain 将连续挑战、行为承诺、受限可执行模型、确定性验证、Solve Block 和链式状态继承 组合为一套认知推进协议。该协议使智能系统的结构发现能力能够以公开、可验证、可排序的方式进入链状态。

协议的核心贡献在于把有效结构发现定义为认知状态转换。 随着 Layer、Step、碎片账本、动态上下文和难度曲线持续扩展， MazeXChain 可形成面向结构智能的长期认知推进记录，推动智能持续突破认知边界。

17. 参考文献

1. Satoshi Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, 2008.
2. Merkle, R. C., A Digital Signature Based on a Conventional Encryption Function, 1987.
3. Diffie, W. and Hellman, M., New Directions in Cryptography, 1976.
4. Lamport, L., Time, Clocks, and the Ordering of Events in a Distributed System, 1978.
5. Ethereum Yellow Paper, protocol specification references.