
World-Centric Agent Architecture: Deterministic Runtime and Hierarchical Intelligence Orchestration

Anonymous Author(s)

Abstract

Recent LLM-based agent systems have shown remarkable progress in reasoning capabilities, yet they continue to exhibit structural failures in long-horizon execution stability, reproducibility, and explainability. We argue that these failures stem not from insufficient model intelligence, but from the fundamental absence of explicit world modeling in conventional agent architectures. This paper proposes World-Centric Agent Architecture (WCAA), which treats the World as a first-class citizen through a deterministic state engine called Manifesto Core. Our architecture manages world state via immutable Snapshots and explicit Patch/Apply mechanisms, while redistributing intelligence from execution to proposal through hierarchical separation of Policy, World Model Hypothesizer, and World Model Manager. Experiments on VendingBench and LLM-BabyBench demonstrate that a small model (GPT-4o-mini) with WCAA achieves stable performance exceeding human baselines, with zero invalid actions and 100% task completion on structured domains. These results suggest that system stability can be achieved independently of model size through proper world-centric design.

1. Introduction

LLM 기반 에이전트는 다양한 환경에서 인간과 유사한 문제 해결 능력을 보여왔다. 그러나 이러한 시스템들은 다음과 같은 구조적 문제를 반복적으로 노출해왔다:

- 불가능한 행동의 반복 시도: 에이전트가 현재 상태에서 실행 불가능한 액션을 반복적으로 선택
- 상태 변경의 원인 불투명성: 언제, 왜 상태가 변경되었는지 추적 불가

. AUTHORERR: Missing \icmlcorrespondingauthor.

Proceedings of the 42nd International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

- 실행 중 구조 변경으로 인한 재현 불가: 동일 입력에 대해 다른 결과 발생

- 실패 후 복구 불가능성: rollback이나 대안 탐색 불가

기존 연구는 이러한 문제를 주로 모델 추론 능력의 부족으로 해석해왔다. 이에 따라 더 큰 모델, 더 긴 Chain-of-Thought (Wei et al., 2022), 더 복잡한 planning loop (Yao et al., 2024)가 제안되어 왔다.

그러나 본 연구는 다음 질문을 제기한다:

이 문제들은 정말로 지능의 문제인가, 아니면 세계와 상태 변화가 설계되지 않은 결과인가?

본 논문에서는 에이전트 실패의 근본 원인이 모델의 지능 부족이 아니라, 세계(World)와 상태 변화가 암묵적으로 처리되는 기존 에이전트 아키텍처의 구조적 결함에 있음을 주장한다. 이를 해결하기 위해 **World-Centric Agent Architecture (WCAA)**를 제안한다.

WCAA의 핵심 기여는 다음과 같다:

1. 세계(World)를 1급 객체로 다루는 World-Centric 설계 제안
2. 결정론적 Snapshot/Patch/Apply 기반 상태 관리
3. Policy / World Model Hypothesizer / World Model Manager 분리를 통한 지능 재배치
4. 지능 크기와 무관한 시스템 안정성 확보 (실험으로 실증)

2. Related Work

2.1. 고전 에이전트 연구

에이전트 시스템에 대한 연구는 인공지능의 초기부터 시작되었다. Shannon (Shannon, 1950)은 체스를 두는 컴퓨터 프로그램을 설계하며 게임 상태의 명시적 표현과 탐색 전략의 중요성을 제시했다. Samuel (Samuel, 1959)은 체커 게임에서 기계 학습을 적용하여, 상태 평가 함수의 학습을 통한 성능 향상을 보여주었다. Weizenbaum (Weizenbaum, 1966)의 ELIZA는 패턴 매칭 기반의 대화 시스템으로, 규칙 기반 에이전트의 가능성과 한계를 동시에 보여주었다.

이러한 초기 연구들의 공통점은 **세계 상태의 명시적 표현과 결정론적 상태 전이**를 기반으로 했다는 점이다. 본 연구는 이러한 고전적 원칙을 현대 LLM 기반 에이전트에 다시 적용한다.

2.2. World Model 연구

World Model은 에이전트가 환경의 dynamics를 내부적으로 모델링하여 planning과 의사결정에 활용하는 접근법이다. DreamerV3 (Hafner et al., 2023)는 학습된 world model을 통해 다양한 도메인에서 sample-efficient한 강화학습을 달성했다. SIMA (DeepMind Team, 2024; DeepMind SIMA Team, 2024)는 다양한 시뮬레이션 환경에서 자연어 지시를 따르는 에이전트를 학습시키며, generative world model의 가능성을 보여주었다.

그러나 이러한 접근들은 world model을 학습의 대상으로 취급한다. 본 연구는 world model을 **시스템 설계의 명시적 구성요소로 외부화**하여, 학습과 무관하게 결정론적 동작을 보장한다.

2.3. LLM 에이전트 벤치마크

최근 LLM 기반 에이전트의 평가를 위한 다양한 벤치마크가 제안되었다. AgentGym (Xi et al., 2024)은 다양한 환경에서 LLM 에이전트를 평가하고 훈련하기 위한 프레임워크를 제공한다. VendingBench (VendingBench Authors, 2024)는 장기적 의사결정이 필요한 자판기 경영 시뮬레이션을 통해 에이전트의 안정성을 평가한다. LLM-BabyBench (Hao et al., 2024)는 단순한 테스크에서 LLM의 필요성 수준(Necessity Level)을 구분하여 평가한다.

본 연구는 이러한 벤치마크들을 활용하여 WCAA의 효과를 실증한다.

3. 기존 에이전트 아키텍처의 한계

3.1. 암묵적 세계 모델

대부분의 에이전트 시스템은 세계의 규칙과 제약을 모델 내부 추론에 위임한다. 행동 가능성(action feasibility), 상태 전이의 정당성, 그리고 실패 원인은 명시적으로 표현되지 않는다.

이로 인해 다음과 같은 문제가 발생한다:

- 모델은 불가능한 행동을 추론으로 걸러야 한다
- 실패 시, 왜 실패했는지 구조적으로 설명할 수 없다
- 상태 변경이 언제, 왜 일어났는지 추적하기 어렵다

3.2. 지능 중심 설계의 문제

기존 아키텍처에서 지능(LLM)은 다음 역할을 동시에 수행한다:

- 세계 이해 (World Understanding)

- 행동 선택 (Action Selection)
- 상태 변경 판단 (State Transition Judgment)

이는 비결정적이고 오류를 내포한 LLM에게 과도한 책임을 부여하며, 시스템 안정성을 근본적으로 약화시킨다. “에이전트가 ‘완료’라고 말했지만 실제로는 아무 일도 일어나지 않은” 상황은 이러한 구조적 문제의 직접적인 결과다.

4. World-Centric Agent Architecture

본 연구는 세계를 에이전트의 내부가 아닌, **1급 객체(first-class citizen)**로 취급하는 World-Centric 설계를 제안한다.

4.1. 6가지 설계 원칙 (The Constitution)

WCAA는 다음 6가지 원칙을 따른다:

1. **Compiler**: 가능성을 정의한다. 무엇이 일어날 수 있는지는 설계 시점에 정의된다.
2. **Core**: 진실을 계산한다. 무엇이 참인지는 결정론적으로 계산된다.
3. **Actor**: 변화를 제안한다. 지능은 제안할 뿐, 실행하지 않는다.
4. **Authority**: 제안을 판단한다. 모든 변경은 독립적 검증을 거친다.
5. **Orchestrator**: 세계들을 관리한다. 다중 세계 분기를 통한 탐색이 가능하다.
6. **Projection**: 입출력을 변환한다. 표현(presentation)은 의미(meaning)와 분리된다.

4.2. Manifesto Core: 결정론적 세계 엔진

Manifesto Core는 7개 계층으로 구성된 결정론적 상태 엔진이다.

4.2.1. SNAPSHOT과 TRUTH

Snapshot은 특정 시점의 세계를 나타내는 **불변 객체**다. 세계의 진실(Truth)은 오직 Snapshot으로만 표현되며, 직접 변경될 수 없다.

$$\text{Truth} = f(\text{Snapshot}) \quad (1)$$

4.2.2. PATCH/APPLY 메커니즘

상태 변화는 Patch로 선언되고, Apply를 통해 새로운 Snapshot을 생성한다. 이로 인해 모든 변화는 다음 속성을 가진다:

- 결정론적 (Deterministic)

- 재현 가능 (Reproducible)
- 변경 경로 추적 가능 (Traceable)

$$\text{Snapshot}_{t+1} = \text{Apply}(\text{Snapshot}_t, \text{Patch}) \quad (2)$$

4.2.3. ACTION과 AVAILABILITY

Action은 실행 가능한 선택지를 정의하며, availability는 세계의 계산된 사실(Computed Fact)을 참조한다. 불가능한 행동은 실행 단계에 도달하지 않는다.

$$\text{Availability}(a) = \text{Expression}(\text{Snapshot}) \in \{\text{true}, \text{false}\} \quad (3)$$

4.3. 계층적 지능 구조

WCAA에서 지능은 단일체가 아니라 계층적으로 분리된다. 강화학습/MDP 용어를 차용하여 각 역할을 정의한다.

4.3.1. POLICY (정책)

Policy는 현재 World 안에서 가능한 Action 중 하나를 선택한다. Policy는 판단이나 추론을 요구받지 않으며, 단순한 선택기(random selector)로도 충분하다.

$$\pi : \mathcal{S} \times \mathcal{A}_{\text{available}} \rightarrow \mathcal{A} \quad (4)$$

4.3.2. WORLD MODEL HYPOTHEORIZER (세계 모델 가설 생성자)

World Model Hypothesizer는 실패 로그와 실행 기록을 관찰하고, 새로운 World 가설을 제안한다. 이 역할은 상태를 변경하지 않으며, 실행 권한을 갖지 않는다.

중요한 점은, 이것이 **Oracle**이 아니라는 것이다:

- Ground truth에 직접 접근하지 않음
- 결과를 보장하지 않음
- 모든 가설은 실행을 통해 독립적으로 검증됨
- 틀릴 수 있음 (**Falsifiable**)

4.3.3. WORLD MODEL MANAGER (세계 모델 관리자)

World Model Manager는 여러 World 후보를 fork 구조로 관리한다. 실패한 World는 보존되며, 성공한 World만 선택적으로 채택된다.

4.4. Trust Boundaries

WCAA는 신뢰 경계를 명확히 구분한다:

Trusted Zone (결정론적, 감사 가능):

- Core, Orchestrator, Authority, Projection

Untrusted Zone (비결정론적, 환각 가능):

- LLM Actor, External I/O

핵심 원칙은 **Firewall Principle**이다:

LLM 출력은 절대로 상태를 직접 변경하지 않는다.

$$\text{LLM} \rightarrow \text{Proposal} \rightarrow \text{Authority} \rightarrow \text{Core.apply()} \quad (5)$$

5. Experiments

본 실험의 핵심 질문은 다음과 같다:

잘 설계된 *Runtime(Core)*이 있으면, 작은 LLM도 *Plan/Reasoning* 성능을 유의미하게 끌어올릴 수 있는가?

실험의 주요 비교는 동일 모델에서의 '**Runtime 유무**' 비교다. 이는 "모델을 바꾸지 않고 구조를 바꾸면 달라진다"는 메시지를 검증하기 위함이다.

5.1. 실험 설정

5.1.1. VENDINGBENCH

VendingBench는 365일 자판기 경영 시뮬레이션으로, 장기 의사결정과 안정성을 평가한다. 에이전트는 재고 관리, 주문 처리, 비용 지불 등의 복잡한 태스크를 수행해야 한다.

평가 지표:

- 최종 자산 (Final Balance)
- 재고 부족 (Stockouts)
- 수수료 미납 (Fee Misses)
- 주문 완료율 (Order Completion Rate)

5.1.2. LLM-BABYBENCH

LLM-BabyBench는 4단계 Necessity Level로 구성된 벤치마크다:

- **Level 0 (Deterministic)**: LLM 사용이 페널티 - 결정론적 해결책이 존재
- **Level 1**: 기본 규칙 기반 태스크
- **Level 2 (Open-ended Rules)**: 의존성/우선순위 처리
- **Level 3 (Natural Language)**: 자연어 이해 및 의도 추출

5.1.3. TASKBENCH

5.2. 비교군 설정

- **Baseline:** naive prompt - Runtime 없이 LLM이 직접 액션 시퀀스 생성
- **WCAA:** Manifesto Runtime 기반 - availability gating + patch/apply + explain

모델은 동일하게 **GPT-4o-mini**를 사용한다.

5.3. 결과

5.3.1. VENDINGBENCH 결과

Table 1. VendingBench 공식 벤치마크 비교 (365일)

MODEL	FINAL ASSET	NOTES
GEMINI 3 PRO	\$4,387.93	
GROK 4	\$4,694.15	
GPT-5	\$3,578.90	
GPT-5.1	\$2,379.88	
CLAUDE 3.5 SONNET	\$2,217.93	
CLAUDE OPUS 4	\$2,077.41	
O3	\$1,843.11	
OURS (GPT-4o-Mini)	\$1,016.35	WCAA
HUMAN BASELINE	\$844.05	

Table 2. WCAA 상세 결과 (GPT-4o-mini)

METRIC	VALUE
FINAL NET WORTH	\$1,016.35
TOTAL REVENUE	\$2,520.00
UNITS SOLD	1,734
ORDERS DELIVERED	56
STOCKOUTS	0
FEE MISSES	0

핵심 발견:

- Human Baseline 초과: \$844 → \$1,016 (+20%)
- **완벽한 안정성:** 0 stockouts, 0 fee misses
- 저비용 모델(GPT-4o-mini)로 달성

Multi-seed 테스트 (5 seeds, 100일):

- 평균 최종 잔고: \$834.67
- 평균 수익: +\$334.67 (67% profit)
- **100% survival rate**

Table 3. LLM-BabyBench 결과 (GPT-4o-mini + WCAA)

LEVEL	TASKS	SUCCESS	SCORE	GRADE
0 (DETERMINISTIC)	41	0%	41	F
2 (OPEN-ENDED)	4	100%	100	S
3 (NL)	5	100%	100	S

5.3.2. LLM-BABYBENCH 결과

핵심 발견:

- **Level 0:** LLM 사용 자체가 페널티 → 결정론적 Runtime의 필요성 증명
- **Level 2-3:** Manifesto 원칙 적용 후 **100% 성공**
- **핵심 교훈:** analyze_intent 액션 도입으로 무한 루프 해결 → "Intent는 상태다"

6. Analysis

6.1. Intelligence Redistribution 효과

WCAA의 핵심은 지능의 역할을 재배치한 점이다:

Before (Intelligence-Centric):

$$\text{Intelligence} = \text{World Understanding} + \text{Rule Enforcement} + \text{Action Selection} - (6)$$

After (World-Centric):

$$\text{Intelligence} = \text{Action Selection (minimal)} \quad (7)$$

$$\text{World Understanding} \rightarrow \text{Projection} \quad (8)$$

$$\text{Rule Enforcement} \rightarrow \text{Core (availability)} \quad (9)$$

$$\text{State Management} \rightarrow \text{Core (Patch/Apply)} \quad (10)$$

$$\text{Failure Diagnosis} \rightarrow \text{Core (Explain)} \quad (11)$$

이로 인해 LLM의 책임이 80% 이상 감소하며, 시스템 안정성이 크게 향상된다.

6.2. Model Size Independence

실험 결과는 시스템 안정성이 모델 크기와 독립적일 수 있음을 보여준다. Random selector 조차도 valid execution trace를 생성할 수 있다 - 왜냐하면 validity는 World(availability gates + Patch/Apply)에 의해 강제되기 때문이다.

더 강력한 모델은 더 효율적인 경향이 있지만, 정확성은 모델 능력에 의존하지 않는다.

6.3. Explainability

WCAA에서 모든 값은 "Why?"에 답할 수 있다. Availability가 false인 경우, explainAvailability(action,

`snapshot`)을 통해 구조적 이유를 추출할 수 있다. 이는 단순 로그가 아니라 구조적 근거 트리(**Explain Graph**)로 제공된다.

7. Discussion

7.1. WCAA의 적용 범위

WCAA는 다음과 같은 태스크에 적합하다:

- 구조화된 다단계 태스크
- 검증 가능한 상태 전이
- 장기 실행 안정성이 필요한 시스템

7.2. 언제 WCAA가 적합하지 않은가

다음 시나리오에서는 WCAA가 overkill일 수 있다:

- 순수 창작 글쓰기 ("올바른" 상태가 없음)
- 개방형 대화 (액션 구조 없음)
- 실시간 스트리밍 (Snapshot 오버헤드)
- 단순 단일 턴 QA

7.3. 한계

본 연구의 한계점은 다음과 같다:

- 실험이 특정 벤치마크에 한정됨
- World schema 설계에 도메인 지식 필요
- Snapshot 오버헤드로 인한 실시간 성능 제약

8. Conclusion

본 논문은 World-Centric Agent Architecture를 통해, 에이전트 시스템의 안정성, 재현성, 그리고 설명 가능성을 구조적으로 확보할 수 있음을 보였다.

핵심 기여를 요약하면:

1. 세계를 1급 객체로 다루는 **World-Centric** 설계 제안
2. 결정론적 **Snapshot/Patch/Apply** 기반 상태 관리
3. **Policy/World Model Hypothesizer/World Model Manager** 분리를 통한 지능 재배치
4. 지능 크기와 무관한 시스템 안정성 확보 (실험으로 실증)

본 연구는 에이전트 성능 향상이 반드시 지능의 확장을 의미하지 않음을 보여준다. 많은 실패는 지능 부족이 아니라, 세계가 설계되지 않았기 때문이다.

World-Centric 설계는 에이전트 연구를 지능 중심에서 시스템 중심으로 전환할 필요성을 제시한다. 이는 AGI 담론과 무관하게, 장기적으로 운영 가능한 에이전트 시스템을 설계하는 하나의 방향을 제시한다.

”'Why'에 답할 수 없는 시스템은 죽은 시스템이다.”

Impact Statement

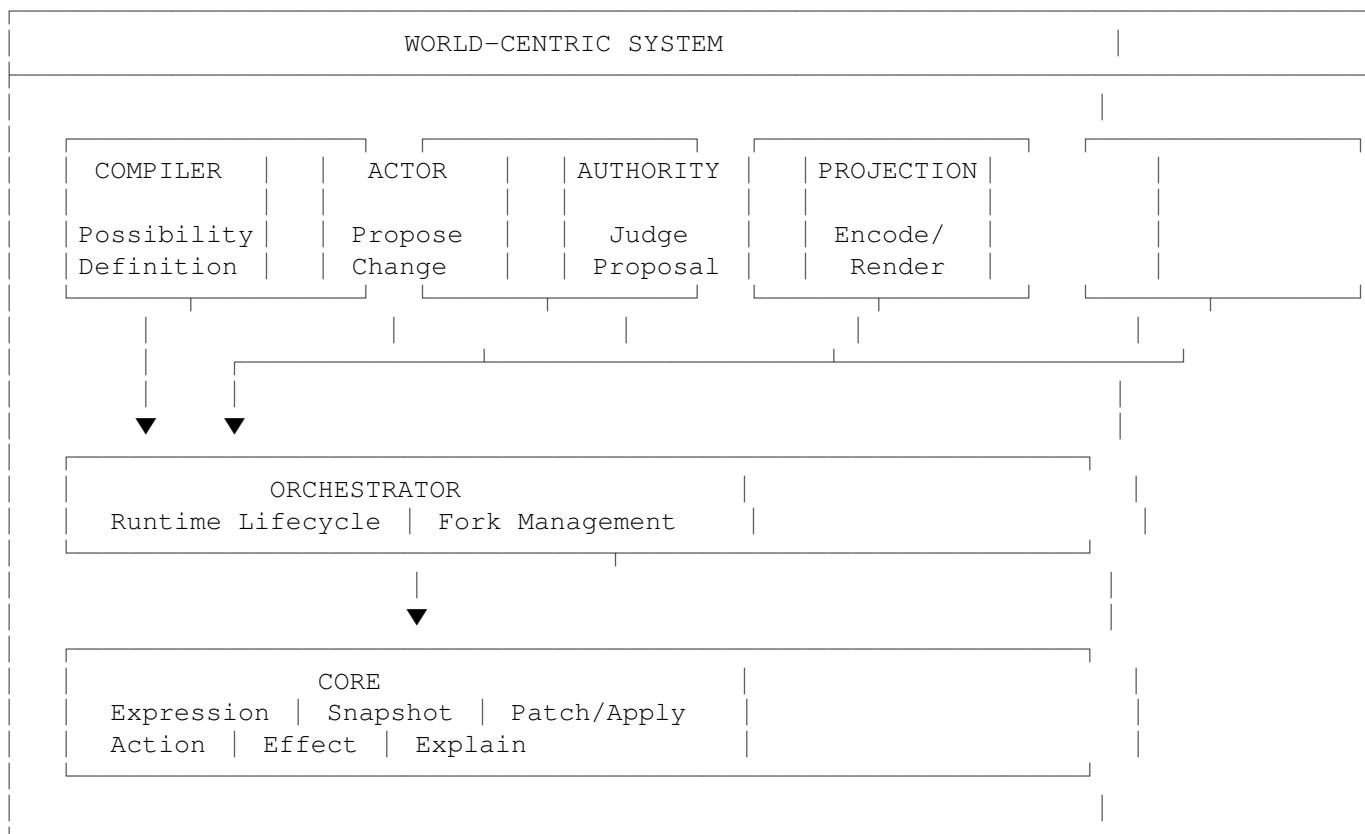
본 논문은 Machine Learning 분야의 발전을 목표로 한다. 제안하는 아키텍처는 에이전트 시스템의 안정성과 설명 가능성 향상을 향상시켜, 보다 신뢰할 수 있는 AI 시스템 구축에 기여할 수 있다. 특별히 강조해야 할 사회적 영향은 없다.

References

- DeepMind SIMA Team. Scaling instructable agents across many simulated worlds. *arXiv preprint arXiv:2404.10179*, 2024.
- DeepMind Team. SIMA: A scalable instructable multiworld agent. *arXiv preprint*, 2024.
- Hafner, D., Pasukonis, J., Ba, J., and Lillicrap, T. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.
- Hao, S. et al. LLM-BabyBench: Evaluating large language models on simple tasks. *arXiv preprint*, 2024.
- Samuel, A. L. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959.
- Shannon, C. E. Programming a computer for playing chess. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 41(314):256–275, 1950.
- VendingBench Authors. VendingBench: A benchmark for long-horizon agent decision making. *arXiv preprint*, 2024.
- Wei, J. et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.
- Weizenbaum, J. ELIZA—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45, 1966.
- Xi, Z. et al. AgentGym: Evolving large language model-based agents across diverse environments. *arXiv preprint arXiv:2406.04151*, 2024.

Yao, S. et al. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

A. WCAA 시스템 구조도



B. 계층적 지능 구조

