BIOPROAGENT: NEURO-SYMBOLIC GROUNDING FOR CONSTRAINED SCIENTIFIC PLANNING

Yuyang Liu[1,*], **Jingya Wang**[1], **Liuzhenghao Lv**[1], **Yonghong Tian**[1,*]
[1]Shenzhen Graduate School, Peking University
*Corresponding to: {liuyuyang13,yhtian}@pku.edu.cn

ABSTRACT

Large language models (LLMs) have demonstrated significant reasoning capabilities in scientific discovery but struggle to bridge the gap to physical execution in wet-labs. In these irreversible environments, probabilistic hallucinations are not merely incorrect but can cause equipment damage or experimental failure. To address this, we propose **BioProAgent**, a neuro-symbolic framework that anchors probabilistic planning in a deterministic Finite State Machine (FSM). We introduce a State-Augmented Planning mechanism that enforces a rigorous *Design-Verify-Rectify* workflow, ensuring hardware compliance before execution. Furthermore, we address the context bottleneck inherent in complex device schemas by *Semantic Symbol Grounding*, reducing token consumption by $\sim 6\times$ through symbolic abstraction. In the extended BioProBench benchmark, BioProAgent achieves 95.6% physical compliance (compared to 21.0% for ReAct), demonstrating that neuro-symbolic constraints are essential for reliable autonomy in irreversible physical environments. Codes release at https://github.com/YuyangSunshine/bioproagent

1 INTRODUCTION

Large Language Models (LLMs) are increasingly shifting from static knowledge retrieval to active world modeling in scientific discovery. In the Self-Driving Laboratories (SDL) (Stein & Gregoire, 2019; Abolhasani & Kumacheva, 2023), LLMs serve as central cognitive engines. These LLMs are categorized as Agentic AI Sapkota et al. (2025); Gridach et al. (2025), design complex chemical synthesis schemes Boiko et al. (2023), discover novel materials Ghafarollahi & Buehler (2025), and act as laboratory assistants Darvish et al. (2025). Recently, multi-agent systems (Jin et al., 2025; Zhou et al., 2025) have demonstrated impressive capabilities in biological reasoning and computational workflow orchestration. However, a critical **Execution Gap** Wang & et al. (2024) emerges when transitioning from computer simulation predictions to in vitro physical experiments. Despite the remarkable physical autonomy achieved by robotic platforms, from mobile robotic chemists Burger et al. (2020) to autonomous materials synthesis labs Szymanski et al. (2023), they still operate under rigid, task-specific workflows. Unlike reversible code and virtual environments in software engineering, due to irreversible physical laws, any deviation in

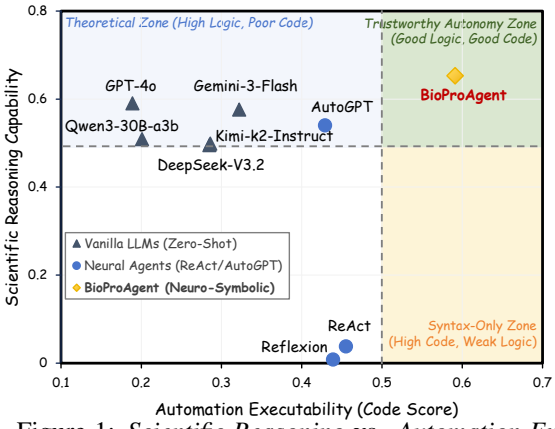


Figure 1: *Scientific Reasoning* vs. *Automation Executability*. **Vanilla LLMs** occupy the *Theoretical Zone* (high logic, weak code), while **Neural Agents (like ReAct)** often generating better code but failing in scientific reasoning. **BioProAgent** achieves *Trustworthy Autonomy* with superior performance in both dimensions.

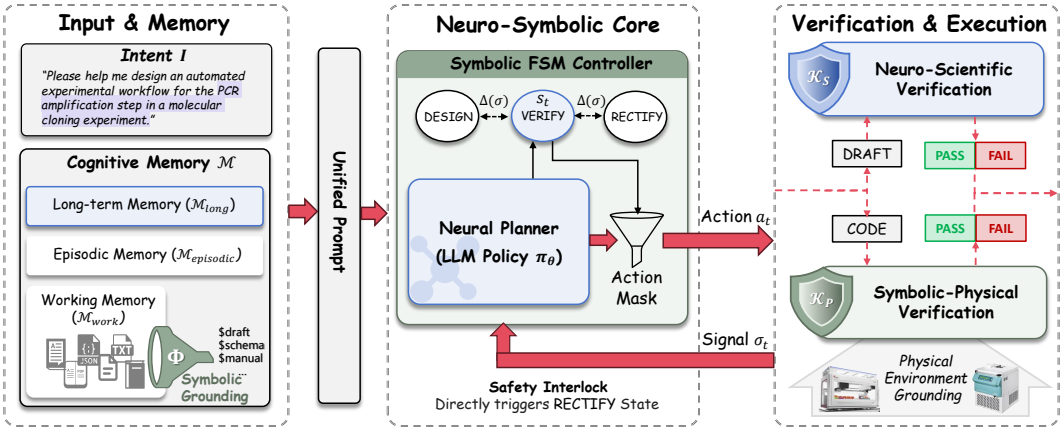


Figure 2: **Overview of BioProAgent.** (1) Cognitive Memory utilizes Symbolic Grounding Φ to manage context efficiently; (2) Neural Planner π_θ is grounded in a Design-Verify-Rectify FSM $\Delta(\sigma)$; (3) Hierarchical Verification $(\mathcal{K}_s, \mathcal{K}_p)$ acts as a safety interlock, enforcing physical compliance by deterministically triggering rectification.

parameters can lead to catastrophic equipment failures, sample loss, or unreproducible results in biological experimental environments.

Despite the reasoning capabilities of existing scientific agents, they still face key challenges in high-risk automated applications. First, standard agents Yao et al. (2022); Shinn et al. (2023) suffer from Cognitive Drift, where contextual overload leads to process confusion, resulting in skipped critical steps or confused temporal dependencies. While current memory systems Chhikara et al. (2025); Packer et al. (2023) excel at semantic retrieval, they struggle with lossless tracking of distinct physical entities. A slight "lost-in-the-middle" Liu et al. (2024) confusion between similar reagent IDs can cause experimental failure. More critically, there is a distinct lack of Pre-Execution Interlocks. While previous neuro-symbolic methods like PAL Gao et al. (2023), Program-of-Thoughts Chen et al. (2022), and SayCan Ahn et al. (2022) have combined LLMs with planners to improve logical correctness, they primarily focus on reasoning optimality. They cannot handle the strict and irreversible safety constraints in wet labs because they lack deterministic control mechanisms based on physical rules that stop and self-correct before execution.

To address these challenges, we propose a neuro-symbolic **BioProAgent** for achieving trustworthy autonomy (Figure 1), which grounds probabilistic reasoning in a deterministic backbone network with a training-free **Finite State Machine (FSM)**. Unlike static agents restricted to pre-defined workflows, BioProAgent flexibly adapts to diverse tasks by using an FSM as a safety boundary to enforce a rigorous "Design-Verify-Rectify" workflow. This controller ensures that all hardware instructions must undergo hierarchical verification for both scientific logic and physical safety before being issued. Furthermore, to address context challenges, we introduce **Semantic Symbol Grounding**. By decoupling the high-dimensional payloads into symbolic pointers, we reduce token consumption by $\sim 6\times$ while ensuring 100% resource consistency. Extensive evaluation on extended BioProBench Liu et al. (2025b) shows that BioProAgent achieves an 88.7% success rate in error recovery, compared to a complete failure (0%) for the standard baseline.

Our contributions are summarized as follows:

- We propose BioProAgent, a training-free neuro-symbolic framework that bridges the gap between probabilistic reasoning and deterministic physical execution.
- We introduce a State-Augmented Planning mechanism driven by a deterministic FSM and a Semantic Symbol Grounding technique that effectively addresses context problem.
- Extensive experiments demonstrate BioProAgent achieves state-of-the-art performance in scientific validity and physical consistency, and significantly outperforms baselines.

2 Related Works

2.1 LLM AGENTS FOR SCIENTIFIC DISCOVERY

Autonomous research has evolved from basic tool usage (Bran et al., 2023; Boiko et al., 2023) to complex multi-agent collaboration (Zhang et al., 2025; Jin et al., 2025; Huang et al., 2025). Recent frameworks like DeepScientist Weng et al. (2025) and Organa Darvish et al. (2025) demonstrate long-horizon autonomy, and BioMARS Qiu et al. (2025) and AutoLabs Panapitiya et al. (2025) introduce domain-specific visual monitoring and self-correction. AI-native operating systems Sim et al. (2024); Fei et al. (2024); Gao et al. (2025) also continuously strengthening the execution infrastructure. However, despite these advances, a critical gap remains: lack cognitive safety interlocking mechanisms, making general-purpose agents vulnerable to irreversible physical damage.

2.2 NEURO-SYMBOLIC REASONING & ACTION

Coupling LLMs with symbolic planners or code interpreters, such as PAL Gao et al. (2023), Program-of-Thoughts Chen et al. (2022), and LLM+P Liu et al. (2023), significantly boosts logical and planning accuracy. In embodied settings, SayCan Ahn et al. (2022) and Voyager Wang et al. (2023) ground actions in physical affordances, while iterative frameworks like ReAct Yao et al. (2022) and Reflexion Shinn et al. (2023) improve reasoning via verbal feedback. While these methods are effective in completing tasks in reversible or simulated environments, they lack mechanisms for enforcing irreversible safety constraints.

2.3 LONG-HORIZON CONTEXT MANAGEMENT

While Longformer Beltagy et al. (2020) and RAG Lewis et al. (2020) extend context windows, they often fragment the coherent narrative essential for protocol execution. Agent-specific memory systems like MemGPT Packer et al. (2023) and Mem0 Chhikara et al. (2025) maintain multi-session coherence through hierarchical compression. Recent approaches such as ACON Kang et al. (2025) and LLM-State Chen et al. (2023) address long-horizon tasks via learned state compression. However, these purely neural methods remain probabilistic and inherently lossy, making them vulnerable to "lost-in-the-middle" phenomenon Liu et al. (2024). This ambiguity is unacceptable for tracking precise scientific entities like reagent IDs.

3 METHODOLOGY

We formulate automated biological protocol generation as a constrained scientific planning problem. Let $\mathcal{P}$ denote possible protocol sequences derived from the valid action space $\mathcal{A}$ defined by the Hardware Registry Ω. Given an intent I and environmental context $\mathcal{E}$, our goal is to identify the optimal executable protocol $P^* \in \mathcal{P}$ that is scientifically valid and physically executable, as:

$$P^* = \underset{P \in \mathcal{P}}{\operatorname{argmax}}\ \mathbb{P}_\theta(P|I) \cdot \mathbb{I}[\mathcal{K}_s(P)] \cdot \mathbb{I}[\mathcal{K}_p(P)], \tag{1}$$

where $\mathbb{I}[\cdot]$ is an indicator function. $\mathcal{K}_s$ and $\mathcal{K}_p$ are the Scientific and Physical verification functions, respectively. (Summary of notations in Appendix A.)

3.1 HYBRID COGNITIVE MEMORY ARCHITECTURE

Existing agents often suffer from cognitive drift in long-horizon tasks, where early context is lost. To address this, we design a hierarchical memory system $\mathcal{M} = \langle \mathcal{M}_{work}, \mathcal{M}_{episodic}, \mathcal{M}_{long} \rangle$, which decouples reasoning logic from high-dimensional data payloads.

Structured Working Memory $\mathcal{M}_{work}$. Direct input of raw JSON schemas (often >10k tokens) induces context saturation. We propose **Semantic Symbol Grounding**, projecting artifacts into symbol-value pairs $\{(k, v)\}$. During planning, context is compressed via a projection function Φ:

$$C_{context} \leftarrow \Phi(\mathcal{M}_{work}) = \{k_i \mid (k_i, v_i) \in \mathcal{M}_{work}\}. \tag{2}$$

For execution, a generated symbolic action a_t is grounded via a resolution function Ψ: $a_t^{exec} \leftarrow \Psi(a_t, \mathcal{M}_{work})$, which maps pointers k back to physical parameters v. This reduces processing complexity from $\mathcal{O}(|v|)$ to $\mathcal{O}(j)$ where $j \ll |v|$, ensuring the planner focuses on logic flow rather than data processing.

Episodic and Long-term Memory $\mathcal{M}_{episodic}$&$\mathcal{M}_{long}$. We use episodic Memory $\mathcal{M}_{episodic}$ to track the active trajectory $\tau_t = [(a_0, o_0), \dots]$ and long-term Memory $\mathcal{M}_{long}$ to retrieve knowledge, ensuring global consistency across different experimental sessions.

The prompt concatenates retrieved knowledge, episodic history, and grounded symbols.

3.2 STATE-AUGMENTED NEURO-SYMBOLIC PLAN

To address the lack of strict adherence to state-related rules required by safety-critical hardware in probabilistic LLMs, BioProAgent uses a deterministic finite state machine (FSM) to decouple reasoning from control, aiming to enforce a rigorous Design-Verify-Rectify (DVR) workflow. Formally defined as $FSM = \langle \mathcal{S}, \Sigma, \mathcal{A}, \Delta, \pi_\theta \rangle$, unlike finite state automata used in traditional robots which hard-code specific operation sequences (e.g., `Move_A_to_B`), our states (e.g., `DESIGN`, `VERIFY`, `RECTIFY`) control the cognitive processes of generation and verification, as illustrated in Fig. 2. This abstraction allows BioProAgent to maintain a consistent control architecture across different domains, from molecular cloning to chemical synthesis, without requiring architectural reconstruction.

3.2.1 NEURO-SYMBOLIC CONTROL ENGINE

Define $\Sigma = \{0, 1\}^5$ as the space of context signals. Let $\sigma_t \in \Sigma$ be the signal at time t. State transitions $\Delta : \mathcal{S} \times \Sigma \to \mathcal{S}$ are governed by a deterministic *Priority Decision Matrix* (Table 1). The physical execution function E_{phys} for an action a_t is defined as:

$$E_{phys}(a_t) = \begin{cases} \text{Execute}(a_t), & \text{if } s_t = \text{APPROVED} \wedge \sigma^+_{phys} \\ \emptyset, & \text{otherwise (Interlock Triggered)} \end{cases} \tag{3}$$

We formalize the system's reliability as a conditional safety guarantee: $P(\text{Unsafe} \mid \text{Interlock}, \mathcal{K}_\Omega, \Phi) \approx 0$. Interlock denotes the FSM gating mechanism. The guarantee holds under two assumptions: (1) $\mathcal{K}_\Omega$: the hardware registry Ω completely covers the physical constraints; and (2) Φ: the semantic symbol grounding correctly maps natural language intents to specific device IDs.

For unmodeled risks outside Ω, the system defaults to a `HALT` state to ensure safety. The FSM evaluates the ordering rules and selects the target state based on the highest-priority signal match:

$$s_{t+1} = \Delta(\sigma_t, s_t) = \text{Target}(r_j) \in s_{t+1} \in \mathcal{S} \tag{4}$$

where $j = \min\{i \mid \text{Match}(r_i, \sigma_t)\}$. As shown in Table 1, if technical validation fails, the interlock strictly prohibits execution. The complete neuro-symbolic execution, coordinating the interaction between $\Delta(\sigma)$ and π_θ, is formalized in Algorithm 1

Algorithm 1 BioProAgent FSM-Gated Execution

Require: User Intent I, Hardwares Ω, Knowledge $\mathcal{M}_{long}$
Ensure: Protocol P^* or Failure
Initialize: $\mathcal{M}_{work} \leftarrow \text{Parse}(I)$, $s_0 \leftarrow$ INIT
Initialize: Trajectory $\tau \leftarrow []$, Signal $\sigma \leftarrow \mathbf{0}$
while $s_t \neq$ SUCCESS **and** $t < T_{max}$ **do**
  $s_t \leftarrow \Delta(\sigma)$ {Refer to Priority Matrix (Tab. 1)}
  if $s_t \in$ {RECTIFY_DRAFT or _CODE} **then**
    Update prompt via σ
  end if
  Generate $a_t \sim \pi_\theta(a|s_t, \Phi(\mathcal{M}_{work}), \tau)$
  if a_t is `Draft` **then**
    Check: $\mathcal{K}_s(a_t)$; Update: σ_{sci}
  else if a_t is `Code` **then**
    Check: $\mathcal{K}_p(a_t, \Omega)$; Update: σ_{phys}
    if $\neg\sigma_{phys}$ **then**
      $\sigma \leftarrow$ INTERLOCK$\{ForceTransition\}$
    else
      Append to P^*; $\tau \leftarrow \tau \cup \{a_t\}$
    end if
  end if
  Update Memory $\mathcal{M}_{episodic}$ with (s_t, a_t, σ)
end while
return P^*

3.2.2 NEURAL REASONING POLICY

In each state s_t, π_θ selects an action a_t from $\mathcal{A}$, the set of available tools (Appendix B), categorized into three functional clusters: (1) **Design** (Generation), (2) **Verify** (Validation), and (3) **Rectify** (Correction). To ensure coherence, the FSM dynamically prunes the action space to $\mathcal{A}_{valid} \subset \mathcal{A}$ based on the active phase in s_t. The planner is conditioned on s_t, and the system prompt explicitly enforces this state-dependent action masking (Appendix H.1).

Table 1: **FSM Priority Decision Matrix.** See Appendix C for the complete definition.

Signal Condition (σ_t)	**Priority**	Target State (s_{t+1})
$\sigma^-_{phys} \wedge \sigma_{code}$	1	RECTIFY_CODE
$\sigma^-_{sci} \wedge \sigma_{draft}$	2	RECTIFY_DRAFT
$\neg\sigma_{draft} \wedge \sigma_{know}$	3	DESIGN_DRAFT
$\sigma_{draft} \wedge \sigma^+_{sci}$	4	DESIGN_CODE
$\sigma_{draft} \wedge \neg\sigma_{sci}$	5	VERIFY_DRAFT
$\sigma_{code} \wedge \sigma^+_{phys}$	6	SUCCESS
...	...	...

Table 2: **The BioProAgent Evaluation Framework.** Metrics are categorized by dimension, improvement direction (↑/↓), and evaluation source.

Dimension	**Evaluator**	**Key Metrics**
Scientific	Semantic Match	S_{sem} (↑), ROUGE-L (↑)
	LLM-as-a-Judge	C_s (↑)
Physical	Rule Engine	S_{code} (↑), C_p (↑)
	Ground Truth	Acc_{seq} (↑), Acc_{param} (↑)
Efficiency	System Logs	$Succ.$ (↑), Tokens (↓), Loop Rate (↓)

3.3 Hierarchical Verification Framework

To implement the **Verify** phase of the DVR loop, we employ a two-layer hierarchical framework to enforce the constraints in Eq. (1).

Layer 1: Neuro-Scientific Verification ($\mathcal{K}_s$). The Scientific Reflector assesses the **Design** draft using prompts derived from expert-in-the-loop iterations. This ensures the CoT reasoning Wei et al. (2022) follows professional wet-lab standards rather than superficial syntax checking.

$$\mathbb{I}[\mathcal{K}_s(P_{draft})] \iff \text{Reflector}(P, \text{Task}) \rightarrow \text{“PASS”}. \tag{5}$$

Full verification criteria, including controls and biosafety checks, are detailed in Appendix H.2.

Layer 2: Symbolic-Physical Verification ($\mathcal{K}_p$). To ensure safety, a deterministic *Rule Engine* ($\mathcal{R}_{phys}$) verifies the generated Code against the hardware registry Ω. Ω encodes constraints for 22 instruments across Liquid Handling (LH), Thermal Control (TC), and Centrifugation (CF). Crucially, any violation ($\neg K_p$) triggers the **Rectify** interlock:

$$\mathbb{I}[\mathcal{K}_p(P_{code})] = \prod_{op \in P} \prod_{c \in \Omega(\text{device}_{op})} \mathbb{I}[\text{Check}(op, c)]. \tag{6}$$

4 Experiments

4.1 Experimental Setup

Benchmark and Hardware Grounding. We evaluate our framework using an extended version of BioProBench Liu et al. (2025b), including four specialized subsets: **Subset A (Protocol Drafting)** assesses scientific validity; **Subset B (Code Generation)** evaluates hardware schema compliance and parameter accuracy; **Subset C (Long-Horizon)** targets global orchestration, featuring 9 complex protocols with 547 atomic steps (max 238 per task); and **Subset D (Error Correction)** measures robustness against injected faults. To ensure reproducibility, the complete source code, the digitized hardware registry (Ω) covering 22 core synthetic biology instruments, and dataset are provided in supplementary material. Although evaluated in simulation, all generated instructions strictly adhere to the API specifications of standard automated devices to bridge the sim-to-real gap.

Baselines. We compare BioProAgent against three categories of baselines: (1) **Vanilla LLMs**: GPT-4o (Islam & Moushi, 2025), Gemini-3-Flash Google (2025), DeepSeek-V3.2 Liu et al. (2025a), and Kimi-k2-instruct Team et al. (2025) via direct prompting; (2) **Standard Agents**: ReAct Yao et al. (2022), Reflexion Shinn et al. (2023), and AutoGPT Yang et al. (2023); (3) **Domain-Specific Agent**: Biomni Huang et al. (2025) (evaluated on native protocol generation). *Critical Implementation Detail:* To ensure fair comparison, all standard agent baselines were equipped with the exact same toolset as BioProAgent.

Evaluation Framework We evaluate performance across three dimensions: (1) **Scientific Validity**: Measured by a composite *Semantic Score* S_{sem} for keyword coverage and an LLM-as-Judge *Scientific Score* C_s for procedural logic. (2) **Physical Compliance**: Quantified by the *Overall Code Score* S_{code}, which aggregates *Physical Compliance* C_p verified by the symbolic rule engine, along

Table 3: Main Results on Protocol Drafting (Subset A) and Code Generation (Subset B). †: indicates statistical significance compared to all baselines (Biomni, ReAct, AutoGPT, Reflexion) with $p < 0.001$ (paired t-test).

		Subset A: Scientific Reasoning				**Subset B: Hardware Execution**		
Method	**Backbone**	ROUGE-L ↑	S_{sem} ↑	C_s† ↑	Time (s) ↓	S_{code} ↑	C_p ↑	Acc_{param} ↑
Direct	GPT-4o	0.107	0.202	0.189	13.8	0.590	0.995	0.295
Direct	Gemini-3-Flash	0.130	0.247	0.322	12.1	0.576	0.996	0.287
Direct	DeepSeek-V3	0.123	0.260	0.285	52.1	0.495	0.995	0.205
Biomni	(Specialized)	0.081	0.252	0.342	87.1	N/A	N/A	N/A
ReAct	Gemini-3-Flash	0.116	0.268	0.455	**44.5**	0.038	0.210	0.103
Reflexion	Gemini-3-Flash	0.118	0.282	0.439	148.4	0.278	0.534	0.403
AutoGPT	Gemini-3-Flash	0.116	0.258	0.429	119.6	0.540	0.911	0.468
BioProAgent	Gemini-3-Flash	**0.147**	**0.344**	**0.591**	71.8	**0.653**	**0.956**	**0.610**

Table 4: Main Results on Long-Horizon Stability (Subset C) and Error Correction (Subset D). Bold indicates the best performance within agent-based frameworks.

		Subset C: Long-Horizon Stability			**Subset D: Error Correction**		
Method	**Backbone**	Succ. ↑	Acc_{param} ↑	C_p ↑	ACC_{seq} ↑	C_p ↑	Loop Rate ↓
ReAct	Gemini-3-Flash	88.9%	0.114	0.217	0.0%	0.000	40.0%
Reflexion	Gemini-3-Flash	33.3%	0.000	0.000	0.0%	0.000	0.0%
AutoGPT	Gemini-3-Flash	66.7%	0.409	0.644	0.0%	0.000	0.0%
BioProAgent	Gemini-3-Flash	**100.0%**	**0.718**	**0.950**	**0.464**	**0.887**	**0.0%**

with *Sequence* Acc_{seq} and *Parameter Accuracy* Acc_{param}. Notably, S_{code} acts as a hard gatekeeper: schema format failures yield a zero score. (3) **System Efficiency**: Tracks *Success Rate* $Succ.$, token consumption *Tokens*, and *Loop Rate*. Table 2 summarizes these metrics; mathematical derivations are detailed in Appendix D.

4.2 Main Results

4.2.1 Scientific Reasoning (Subset A).

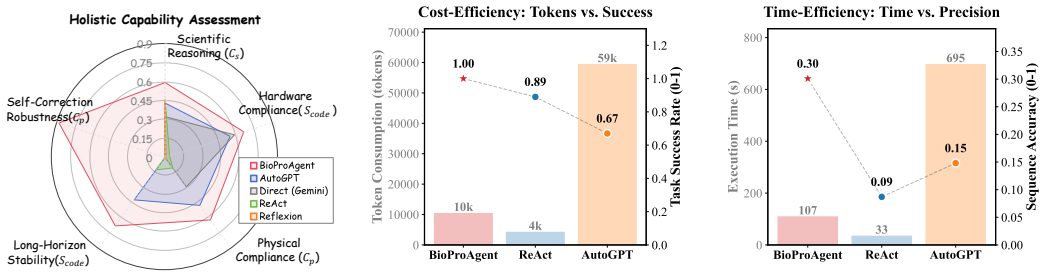
As shown in Table 3, BioProAgent achieves a Scientific Validity Score (C_s) of 0.591, a **30% improvement** over the strongest agent baseline (ReAct) and almost double that of vanilla models (Gemini-3-Flash). Notably, even the domain-specialized Biomni only reaches a C_s of 0.342, highlighting that domain knowledge alone is insufficient without structured reasoning constraints. Paired t-tests confirm that these improvements are statistically significant ($p < 0.001$). BioProAgent's advantage stems from its *Scientific Reflector* within the FSM, which requires a formal review of scientific logic before proceeding to the coding phase, whereas standard agents often neglect crucial controls in pursuit of rapid generation. Cross-model validation (Pearson $r \geq 0.84$) confirms the robustness of our LLM-as-a-Judge metric (see Appendix D.4.2).

4.2.2 Physical Compliance (Subset B).

Table 3 reveals significant differences in code generation performance between the baselines. Vanilla LLMs maintain $C_p = 0.995$ primarily by defaulting to conservative, low-complexity code, which naturally results in poor Parameter Accuracy (0.295). Standard Agents (e.g., ReAct), conversely, attempt complex operations but suffer from catastrophic safety failures ($C_p = 0.210$). *Although ReAct was equipped with the exact same Rule Engine tool, it often fails to invoke these tools proactively due to context saturation or a lack of stopping conditions.* BioProAgent effectively eliminates this trade-off, achieving the highest S_{code} and ACC_{param} while maintaining high-quality security ($C_p = 0.956$). This confirms FSM's deterministic gating is essential for enforcing safety checks that probabilistic agents tend to bypass.

4.2.3 Long-Horizon Stability (Subset C).

As shown in Table 4, standard agents exhibit significant performance gaps when the workflow expands to an average of 60 steps. While AutoGPT maintains a 66.7% success rate, its lack of a



(a) (b) (c)

Figure 3: **Efficiency and Reliability Analysis.** (a) **Multidimensional Capabilities:** BioProAgent (red) achieves the most expansive capability envelope across five critical dimensions. Note that identical metrics on different axes represent performance in distinct evaluation subsets. (b) **Cost-Efficiency:** On long-horizon tasks (Subset C), our system reduces token consumption by 82% compared to AutoGPT while maintaining a 100% success rate. (c) **Time-Efficiency:** BioProAgent demonstrates superior precision with concise execution time.

structural stopping condition leads to inefficient thought loops and excessive token consumption. ReAct achieves an 88.9% task completion rate, but its low physical consistency ($C_p = 0.217$) and $Acc_{param} = 0.114$ reveal ID drift. *Notably, Reflexion fails completely in* $Acc_{param} = 0.000$. *This is attributed to "Schema Corruption": its verbal feedback mechanism frequently injects conversational artifacts (e.g., apologies) into the JSON payload, causing the strict rules engine to fail to parse the output.* In contrast, by replacing the high-dimensional data payloads with symbolic pointers, our *Semantic Symbol Grounding* ensures 100% resource consistency and lossless state tracking.

4.2.4 Error Correction (Subset D).

Subset D evaluates the system's ability to diagnose and correct injected physical violations, as shown in Table 4. All baseline agents exhibit a 0% correction rate, with ReAct entering infinite retry loops in 40% of scenarios. This shows that without a deterministic controller, probabilistic LLMs naturally prioritize sequence plausibility over physical safety. BioProAgent restores physical safety (C_p) to 0.887, significantly outperforming the baselines (0.0). The neuro-symbolic FSM acts as a deterministic router: when a violation is detected by the rules engine, it overwrites the LLM's trajectory and forces a transition to the `RECTIFY_CODE` state. This allows the agent to align its outputs with hardware limitations, demonstrating a robust self-correcting capability essential for trustworthy automated science. Appendix E.2 provides a detailed analysis of logical errors, resource errors, and physical errors.

4.3 Holistic Capability Assessment

Fig. 3a visualizes the capability envelope of each agent. Crucially, this chart aggregates performance across diverse benchmarks: while C_p and S_{code} appear on multiple axes, they correspond to distinct stress tests, standard generation (Subset B), long-horizon stability (Subset C), and error correction (Subset D). Existing methods exhibit significant capability biases: Direct Prompting (Grey) defaults to conservative safety (C_p) but fails in complex coordination, while ReAct (Green) improves scientific reasoning (C_s) at the cost of severe physical hallucinations. BioProAgent effectively eliminates this trade-off, achieving a balanced pentagonal coverage. The performance gap is most significant in Long-Horizon and Self-Correction dimensions. While baselines degrade to near-zero performance due to context collapse or open-loop error propagation, BioProAgent maintains robustness, demonstrating that neuro-symbolic FSM is essential for sustaining trustworthy autonomy in high-stakes environments.

4.4 Efficiency and Cost Analysis

As shown in Fig. 3(b), BioProAgent consumes **82% fewer tokens** than the AutoGPT on Subset C. This efficiency stems from our *Semantic Symbol Grounding* mechanism, which maintains a near-constant context size by substituting dense data payloads with symbolic pointers, thereby mitigating

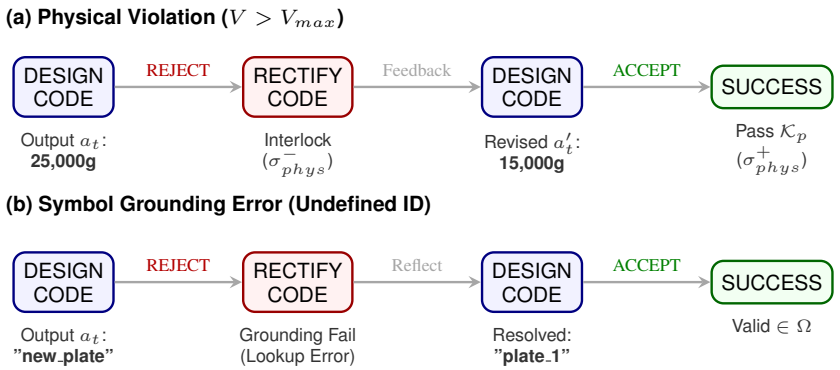


Figure 4: **FSM-Driven Self-Correction Trajectories.** (a) Physical Correction. (b) Symbol Grounding Error.

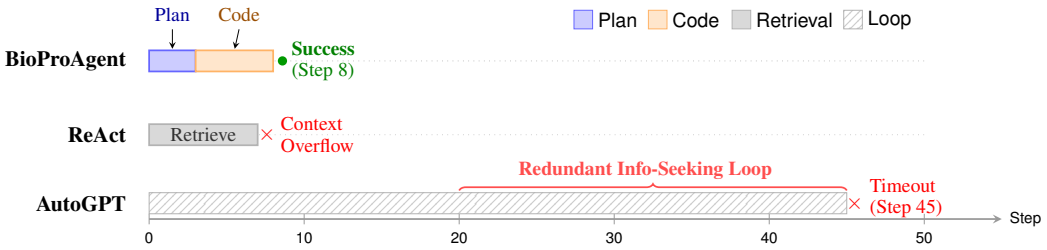


Figure 5: **Trace Analysis (Subset C).** Execution divergence on a long-horizon task. TOP: BioProAgent efficiently transitions to coding via FSM. MIDDLE: ReAct fails early due to context overflow. BOTTOM: AutoGPT wastes resources in a retrieval loop.

the "context explosion" inherent in the recursive summarization. Furthermore, Fig. 3(c) highlights the critical trade-off between execution time and precision. While standard baselines are faster, they lack the requisite precision for complex hardware operations. While the AutoGPT attains moderate precision, its execution time is prohibitively long (avg. 695s). BioProAgent achieves an optimal balance, delivering high-precision results with competitive execution speeds and the widest safety margin (C_p), validating practical viability for the real-world laboratory deployment.

4.5 Qualitative Analysis

Mechanism of Correction. Standard LLM agents typically operate in an open-loop manner: when a dangerous parameter for an hallucinate is generated, execution is immediate, inevitably leading to failure. In contrast, Fig. 4 illustrates the FSM's correction mechanism in action, effectively intercepting speed limit violations. In the physical violation case (Fig. 4a), the *Symbolic Rule Engine* ($\mathcal{K}_p$) preemptively intercepts this violation ($V > V_{max}$), triggering a forced state transition to `RECTIFY_CODE`. This forces the Neural Planner to recognize the specific error signal (σ^-_{phys}) and regenerate the code within safe limits (15,000g). Similarly, for a logical hallucination case (Fig. 4b), the system detectes semantic drift (an undefined resource ID "`new_plate`"). The FSM prevents this error propagation, guiding the agent to remap the action to the grounded symbol "`plate_1`".

Log-Level Behavioral Analysis. To diagnose why baseline agents fail in long-horizon tasks despite having access to the same tools, we visualize the execution trajectories of a representative metabolomics workflow in Fig. 5. ReAct terminates prematurely (Step 7) as retrieved API schemas saturate its context window, blocking valid action generation. Lacking a deterministic *stop* signal, AutoGPT enters an infinite loop of redundant `retrieve_knowledge` calls (Steps 20–45), consuming excessive tokens without advancing state. In contrast, BioProAgent's FSM enforces a rigid *Plan* $\rightarrow$ *Code* transition, acting as a cognitive clock that ensures efficient completion in just 8 steps.

Table 5: **Representative Error Corrections in Subset D.** The FSM interceptor enforces the **Rectify** phase, guiding the agent from violations to valid execution.

Constraint	**Design Violation** (s_{design})	**FSM Rectification** ($s_{rectify}$)
Physical ($\mathcal{K}_p$) *Trace*	`centrifuge(speed='25000g')` × Exceeds rotor limit (15k)	`centrifuge(speed='15000g')` ✓ Clamped to safety max
Symbol (Φ) *Trace*	`transfer(source='buffer_x')` × Grounding Fail (No ID)	`transfer(source='trough_1')` ✓ Remapped to registry

Causal
 	
Trace
	`seal_plate();`
	`transfer(...)`
	× Blocked op (Sealed)
		`transfer(...);`
		`seal_plate()`
		✓ Reordered dependency

Table 6: Ablation Study of BioProAgent Components.

Variant	**Subset A** C_s	**Subset B** C_p	**Subset C** S_{code}	**Subset D** C_p
Full Model	**0.591**	**0.956**	**0.668**	**0.887**
w/o FSM	0.262	0.049	0.000	0.000
w/o Verification	0.563	0.902	0.648	0.870
w/o Knowledge	0.561	0.889	0.657	N/A
w/o Clarify	0.403	N/A	0.632	N/A

Granularity of Rectification. Table 5 further demonstrates that these corrections are not merely stochastic retries but involve semantic-level reasoning. We observe three distinct categories of rectification within the DVR loop: (1) **Safety Alignment:** Adjusting physics parameters to satisfy hardware constraints ($\mathcal{K}_p$); (2) **Resource Grounding:** Resolving hallucinated variables to physically validated slots via the projection function Φ; and (3) **Causal Rectification:** Correcting logical dependencies, such as the critical reordering of `seal_plate()` operations. This confirms that BioProAgent's neuro-symbolic core effectively operationalizes a "Design-Verify-Rectify" workflow, ensuring trustworthy execution in high-stakes environments.

4.6 Ablation Study

Table 6 quantitatively validates the architectural necessity of each component. **Impact of FSM:** Removing the FSM controller leads to a structural failure. The Scientific Validity (C_s) degrades to 0.262, while performance on long-horizon (Subset C) and error-correction (Subset D) tasks collapses to near-zero (0.000). This confirms that deterministic state transitions are not merely auxiliary, but foundational for sustaining the planning-execution loop and triggering self-healing mechanisms. **Impact of Verification:** Ablating the verifications ($\mathcal{K}_p, \mathcal{K}_s$) causes C_p to drop from 0.956 to 0.902. While less dramatic than the FSM ablation, this deficit demonstrates that neural inference alone, even with advanced prompting, cannot guarantee the zero-defect safety required for irreversible wet-lab automation. **Impact of Knowledge & Clarification:** Omitting the *Clarify* tool or external knowledge significantly impairs reasoning. Notably, the sharp decline in C_s ($0.591 \rightarrow 0.403$) without clarification underscores the critical role of human-in-the-loop disambiguation for handling under-specified protocols. Statistical significance is further confirmed via cross-judge verification (Appendix D.4.1) and paired t-tests (Appendix D.4.2).

5 Conclusion

We present BioProAgent, a neuro-symbolic framework addressing the critical execution gap in autonomous scientific discovery. By grounding probabilistic LLM reasoning within a deterministic Finite State Machine, BioProAgent enforces a "Design-Verify-Rectify" workflow for irreversible wet-lab environments. Evaluations on extended BioProBench demonstrate that BioProAgent achieves SOTA performance, showcasing robust autonomous self-correction where traditional agents fail. Our work emphasizes that for high-risk physical applications, neural intelligence must be constrained by symbolic safety interlocks to ensure trustworthy and reliable autonomy.

LIMITATIONS & FUTURE WORK

Despite its performance, BioProAgent has some limitations. First, the framework's security relies on a predefined hardware registry (Ω). While effective for known instruments, this requires manual registration for custom hardware. Future work will explore using a multimodal logic model to automate this process by directly reading the device manual. Additionally, our current evaluation was conducted in high-fidelity simulations using standard APIs. While this validates the correctness of the logic and parameters, real-world physical randomness remains unmodeled. Integrating a real-time visual feedback loop is crucial for managing these physical variables.

ACKNOWLEGEMENT

This work was supported by the China Postdoctoral Science Foundation under Grant Number BX20240013 and 2024M760113.

REFERENCES

Milad Abolhasani and Eugenia Kumacheva. The rise of self-driving labs in chemical and materials sciences. *Nature Synthesis*, 2(6):483–492, 2023.

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Daniil A Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, 2023.

Andres M Bran, Sam Cox, Andrew D White, and Philippe Schwaller. Chemcrow: Augmenting large-language models with chemistry tools. *arXiv preprint arXiv:2304.05376*, 2023.

Benjamin Burger, Phillip M Maffettone, Vladimir V Gusev, Catherine M Aitchison, Yang Bai, Xiaoyan Wang, Xiaobo Li, Ben M Alston, Buyi Li, Rob Clowes, et al. A mobile robotic chemist. *Nature*, 583(7815):237–241, 2020.

Siwei Chen, Anxing Xiao, and David Hsu. Llm-state: Open world state representation for long-horizon task planning with large language model. *arXiv preprint arXiv:2311.17406*, 2023.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*, 2022.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

Kourosh Darvish, Marta Skreta, Yuchi Zhao, Naruki Yoshikawa, Sagnik Som, Miroslav Bogdanovic, Yang Cao, Han Hao, Haoping Xu, Alán Aspuru-Guzik, Animesh Garg, and Florian Shkurti. Organa: A robotic assistant for automated chemistry experimentation and characterization, 2025. URL `https://arxiv.org/abs/2401.06949`.

Yuxing Fei, Bernardus Rendy, Rishi Kumar, Olympia Dartsi, Hrushikesh P Sahasrabuddhe, Matthew J McDermott, Zheren Wang, Nathan J Szymanski, Lauren N Walters, David Milsted, et al. Alabos: a python-based reconfigurable workflow management framework for autonomous laboratories. *Digital Discovery*, 3(11):2275–2288, 2024.

Jing Gao, Junhan Chang, Haohui Que, Yanfei Xiong, Shixiang Zhang, Xianwei Qi, Zhen Liu, Junjie Wang, Qianjun Ding, Xinyu Li, et al. Unilabos: An ai-native operating system for autonomous laboratories. *arXiv preprint arXiv:2512.21766*, 2025.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *International Conference on Machine Learning*, pp. 10764–10799. PMLR, 2023.

Alireza Ghafarollahi and Markus J Buehler. Sciagents: automating scientific discovery through bioinspired multi-agent intelligent graph reasoning. *Advanced Materials*, 37(22):2413523, 2025.

Google. Gemini 3 flash model card. Online, December 2025. URL https://storage.googleapis.com/deepmind-media/gemini/Gemini-3-Flash-Model-Card.pdf.

Mourad Gridach, Jay Nanavati, Khaldoun Zine El Abidine, Lenon Mendes, and Christina Mack. Agentic ai for scientific discovery: A survey of progress, challenges, and future directions. *arXiv preprint arXiv:2503.08979*, 2025.

Maarten Grootendorst. Keybert: Minimal keyword extraction with bert., 2020. URL https://doi.org/10.5281/zenodo.4461265.

Kexin Huang, Serena Zhang, Hanchen Wang, Yuanhao Qu, et al. Biomni: A general-purpose biomedical ai agent. *bioRxiv*, 2025. doi: 10.1101/2025.05.30.656746. Preprint.

Raisa Islam and Owana Marzia Moushi. Gpt-4o: The cutting-edge advancement in multimodal llm. In *Intelligent Computing-Proceedings of the Computing Conference*, pp. 47–60. Springer, 2025.

Ruofan Jin, Yucheng Guo, Yuanhao Qu, Ming Yang, et al. Biolab: End-to-end autonomous life sciences research with multi-agents system integrating biological foundation models. *bioRxiv*, 2025. doi: 10.1101/2025.09.03.674085. Preprint.

Minki Kang, Wei-Ning Chen, Dongge Han, Huseyin A Inan, Lukas Wutschitz, Yanzhi Chen, Robert Sim, and Saravan Rajmohan. Acon: Optimizing context compression for long-horizon llm agents. *arXiv preprint arXiv:2510.00615*, 2025.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025a.

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shuo Zhang, Joydeep Biswas, and Peter Stone. Llm+p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*, 2023.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

Yuyang Liu, Liuzhenghao Lv, Xiancheng Zhang, Li Yuan, and Yonghong Tian. Bioprobench: Comprehensive dataset and benchmark in biological protocol understanding and reasoning. *arXiv preprint arXiv:2505.07889*, 2025b.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G Patil, Ion Stoica, and Joseph E Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Gihan Panapitiya, Emily Saldanha, Heather Job, and Olivia Hess. Autolabs: Cognitive multi-agent systems with self-correction for autonomous chemical experimentation. *arXiv preprint arXiv:2509.25651*, 2025.

Yibo Qiu, Zan Huang, Zhiyu Wang, Handi Liu, Yiling Qiao, Yifeng Hu, Shu'ang Sun, Hangke Peng, Ronald X Xu, and Mingzhai Sun. Biomars: A multi-agent robotic system for autonomous biological experiments. *arXiv preprint arXiv:2507.01485*, 2025.

Ranjan Sapkota, Konstantinos I Roumeliotis, and Manoj Karkee. Ai agents vs. agentic ai: A conceptual taxonomy, applications and challenges. *arXiv preprint arXiv:2505.10468*, 2025.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

Malcolm Sim, Mohammad Ghazi Vakili, Felix Strieth-Kalthoff, Han Hao, Riley J Hickman, Santiago Miret, Sergio Pablo-García, and Alán Aspuru-Guzik. Chemos 2.0: An orchestration architecture for chemical self-driving laboratories. *Matter*, 7(9):2959–2977, 2024.

Helge S Stein and John M Gregoire. Progress and prospects for accelerating materials science with automated and autonomous workflows. *Chemical science*, 10(42):9640–9649, 2019.

Nathan J Szymanski, Bernardus Rendy, Yuxing Fei, Rishi E Kumar, Tanjin He, David Milsted, Matthew J McDermott, Max Gallant, Ekin Dogus Cubuk, Amil Merchant, et al. An autonomous laboratory for the accelerated synthesis of inorganic materials. *Nature*, 624(7990):86, 2023.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Xiaoxuan Wang and et al. Scibench: Evaluating college-level scientific problem-solving abilities of large language models. *arXiv preprint arXiv:2307.10635*, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Yixuan Weng, Minjun Zhu, Qiujie Xie, Qiyao Sun, Zhen Lin, Sifan Liu, and Yue Zhang. Deepscientist: Advancing frontier-pushing scientific findings progressively. *arXiv preprint arXiv:2509.26603*, 2025.

Hui Yang, Sifu Yue, and Yunzhong He. Auto-gpt for online decision making: Benchmarks and additional opinions. *arXiv preprint arXiv:2306.02224*, 2023.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022.

Zhen Zhang, Zhichu Ren, Chia-Wei Hsu, Weibin Chen, Zhang-Wei Hong, Chi-Feng Lee, Aubrey Penn, Hongbin Xu, Daniel J Zheng, Shuhan Miao, et al. A multimodal robotic platform for multi-element electrocatalyst discovery. *Nature*, pp. 1–3, 2025.

Yuyang Zhou, Jin Su, Jiawei Zhang, Wangyang Hu, Tianli Tao, Guanqi Li, Xibin Zhou, Li Fan, and Fajie Yuan. Prime: A multi-agent environment for orchestrating dynamic computational workflows in protein engineerings. *bioRxiv*, pp. 2025–09, 2025.

APPENDIX

A TABLE OF NOTATIONS

Table 10 provides a comprehensive reference for the mathematical notations and symbols employed throughout this paper. To facilitate readability, the symbols are categorized by their functional roles within the BioProAgent framework: **Problem Formulation**, **Cognitive Memory**, and the **Neuro-Symbolic Core**.

B DETAILED TOOLSET

To ensure fairness and reproducibility, all agents in our experiments are equipped with the identical set of operational tools. Table 11 details the specifications, categorized by their functional role in the **Design-Verify-Rectify (DVR)** workflow.

C FSM DECISION LOGIC

The Finite State Machine (FSM) acts as the deterministic backbone of BioProAgent, orchestrating the DVR loop. Table 7 details the **Priority Decision Matrix** used to compute state transitions $s_{t+1} = \Delta(\sigma_t)$. The rules are evaluated in descending order of priority (1 is highest), ensuring that safety interlocks always override generative actions.

Table 7: **The Priority Decision Matrix defining the FSM transition logic.** σ^-_{phys} denotes a physical validation failure, while σ^+_{sci} denotes a scientific validation success. Note how the **[Rectify]** interlocks (Rows 1-2) take strict precedence over **[Design]** steps.

Priority	Signal Condition (σ_t)	Target State (s_{t+1})	Logic Description (DVR Flow)
1	$\sigma^-_{phys} \wedge \sigma_{code}$	RECTIFY_CODE	**[Rectify] Safety Interlock**: If Rule Engine ($\mathcal{K}_p$) detects a physical violation (Severity: HALT), immediately force a fix.
2	$\sigma^-_{sci} \wedge \sigma_{draft}$	RECTIFY_DRAFT	**[Rectify] Scientific Review**: If Reflector ($\mathcal{K}_s$) rejects the protocol logic (e.g., missing controls), require revision.
3	$\neg\sigma_{draft} \wedge \sigma_{know}$	DESIGN_DRAFT	**[Design] Knowledge Ready**: If knowledge is retrieved but no draft exists, enter drafting mode.
4	$\sigma_{draft} \wedge \sigma^+_{sci}$	VERIFY_DRAFT	**[Verify] Draft Approved**: Mark draft as valid; transition to automation alignment.
5	$\sigma_{approved} \wedge \neg\sigma_{code}$	DESIGN_CODE	**[Design] Code Generation**: Synthesize executable machine code from the verified draft.
6	$\sigma_{code} \wedge \sigma^+_{phys}$	SUCCESS	**[Finish] Completion**: Code exists and passes all physical validations ($\mathcal{K}_p$). Task complete.
7	$\sigma_{ambiguous}$	CLARIFY_INTENT	**[Aux] Ambiguity Check**: If initial intent is unclear, loop back to ask user for parameters.
8	(Default)	s_t (Self-loop)	**[Wait]**: If no condition matches, maintain current state (e.g., during retrieval or accumulation).

D DETAILED EVALUATION METRICS

To rigorously assess the quality of generated protocols and executable machine code, we implement a hybrid evaluation framework that integrates semantic similarity analysis, deterministic rule-based verification, and expert-aligned LLM-as-Judge assessments.

D.1 PROTOCOL GENERATION METRICS

Let $P_{gt} = \{p_1, ..., p_k\}$ denote the ground truth protocol sequence, and $P_{gen} = \{p'_1, ..., p'_m\}$ denote the generated sequence.

Text and Semantic Metrics. We employ three complementary metrics to measure lexical coverage and structural alignment:

1. Keyword F1 ($F1_K$): Utilizing KeyBERT Grootendorst (2020), we extract the top-k ($k = 64$) domain-specific entities (e.g., reagents, instruments). Precision (P_K) and Recall (R_K) are computed based on the entity overlap between P_{gt} and P_{gen}:

$$F1_K = \frac{2 \cdot P_K \cdot R_K}{P_K + R_K}. \tag{7}$$

2. Step Recall (SR) Liu et al. (2025b): To quantify structural completeness, we measure the proportion of ground truth steps that are semantically recovered in the generation:

$$SR = \frac{\sum_{s \in P_{gt}} \mathbb{I}\left[\max_{s' \in P_{gen}} \text{Sim}(s, s') \geq \tau\right]}{|P_{gt}|}, \tag{8}$$

where $\text{Sim}(\cdot)$ denotes the cosine similarity of sentence embeddings derived from `all-mpnet-base-v2`, with a semantic threshold $\tau = 0.7$.

3. Semantic Consistency Score (S_{sem}): We implement regex-based extraction to isolate experimental verbs ($\mathcal{V}$) and numerical parameters ($\mathcal{PR}$). The component scores are defined as:

$$Cov_{\text{step}} = \frac{|\mathcal{V}_{gt} \cap \mathcal{V}_{gen}|}{|\mathcal{V}_{gt}|}, \quad E_{\text{param}} = \frac{|\mathcal{PR}_{gt} \cap \mathcal{PR}_{gen}|}{|\mathcal{PR}_{gt}|}. \tag{9}$$

To ensure numerical stability, if $|\mathcal{V}_{gt}| = 0$ or $|\mathcal{PR}_{gt}| = 0$, the respective sub-score defaults to 0.5. The aggregated semantic score is computed as:

$$\begin{aligned} S_{sem} = 0.15 \cdot Cov_{\text{step}} + 0.15 \cdot E_{\text{param}} \\ + 0.30 \cdot F1_K + 0.40 \cdot SR. \end{aligned} \tag{10}$$

Rationale for Weights: These coefficients were calibrated based on consultations with senior wet-lab biologists to prioritize structural completeness (SR) and keyword accuracy ($F1_K$) over minor syntactic variations. This ensures the metric reflects practical laboratory utility rather than mere linguistic fluency.

Scientific Validity (C_s). Since semantic overlap may miss subtle logical flaws, we leverage an LLM-as-Judge (Gemini-3-Flash) to audit protocols across five weighted dimensions: Critical Step Coverage ($S_{cov}, w_1 = 0.3$), Parameter Precision ($S_{param}, w_2 = 0.2$), Control Fidelity ($S_{ctrl}, w_3 = 0.15$), Biosafety ($S_{safe}, w_4 = 0.1$), and Objective Alignment ($S_{obj}, w_5 = 0.25$).

$$\begin{aligned} C_s = 0.3S_{cov} + 0.2S_{param} + 0.15S_{ctrl} \\ + 0.1S_{safe} + 0.25S_{obj}. \end{aligned} \tag{11}$$

D.2 MACHINE CODE GENERATION METRICS

Let S_{gt} and S_{gen} represent the sequences of executable operations. Code quality is evaluated through a three-layered approach:

Rule-Based Validity. We strictly verify basic executability via deterministic checks:

1. Schema Compliance ($Schema_{comp}$): The proportion of mandatory fields (e.g., node types, connection IDs) that adhere to the JSON definition.

2. Resource Validity (Res_{valid}): The ratio of valid instrument IDs referenced in the code:

$$Res_{valid} = \frac{1}{M} \sum_{i=1}^{M} \mathbb{I}\left(n_i^{\text{id}} \in \Omega_{valid} \cup \{-1\}\right), \tag{12}$$

Table 8: **Multi-Judge Consistency Analysis.** Comparison of C_s scores across three distinct LLMs ($N = 640$ evaluations). GPT-4o confirms the trends with the highest correlation, validating the robustness of our metric.

Model Variant	Gemini	Kimi	GPT-4o
BioProAgent-Full	0.591	0.597	0.515
w/o Knowledge	0.561	0.552	0.497
w/o Clarify	0.403	0.421	0.374
w/o FSM	0.262	0.285	0.105
Correlation (r)w/ Gemini	**1.00**	**0.84**	**0.91**

Table 9: **Statistical Significance Analysis (Paired t-test).** Comparison of BioProAgent against baselines and ablation variants ($N = 160$). All improvements are highly statistically significant.

Category	Comparison Pair (vs. Full Model)	Mean Diff.	P-value
Baselines	ReAct	+0.136	$< 10^{-10}$
	AutoGPT	+0.162	$< 10^{-11}$
	Reflexion	+0.152	$< 10^{-11}$
Ablations	w/o Knowledge	+0.030	0.025^*
	w/o Clarify	+0.188	$< 10^{-13}$
	w/o FSM	+0.329	$< 10^{-49}$

* Significant ($p < 0.05$);
All others are extremely significant ($p < 0.001$).

where Ω_{valid} represents the authorized hardware registry and -1 denotes manual steps.

3. Sequence Accuracy (Acc_{seq}): The Longest Common Subsequence (LCS) ratio measuring the ordering correctness of functional operations:

$$Acc_{seq} = \frac{2 \cdot |\text{LCS}(S_{gt}, S_{gen})|}{|S_{gt}| + |S_{gen}|}. \tag{13}$$

Physical Compliance (C_p). A symbolic rule engine validates N hardware constraints (e.g., $rpm \leq 15,000$). The score penalizes Critical Errors (HALT, $\alpha = 0.2$) and Warnings (WARN, $\beta = 0.05$):

$$C_p = \max\left(0, 1 - \frac{\alpha N_{halt} + \beta N_{warn}}{N_{total}}\right). \tag{14}$$

This penalty mechanism ensures that safety violations significantly degrade the score, aligning with the zero-tolerance policy for hardware damage.

Parameter Accuracy (Acc_{param}). Fine-grained parameter correctness is evaluated by LLM-as-Judge across Node Completeness ($S_{n\text{-}com}$), Value Accuracy ($S_{p\text{-}acc}$), and Field Completeness ($S_{p\text{-}com}$):

$$Acc_{param} = 0.3S_{n\text{-}com} + 0.4S_{p\text{-}acc} + 0.3S_{p\text{-}com}. \tag{15}$$

Overall Code Score (S_{code}). We employ a multiplicative formulation where $Schema_{comp}$ acts as a soft gating factor. Unlike additive metrics where accurate parameters might mask a broken syntax, this design ensures that if the foundational schema structure is flawed ($Schema_{comp} \to 0$), the overall executability score S_{code} is penalized towards zero. This reflects the operational reality that syntactically invalid code is non-executable regardless of semantic intent.

$$\begin{aligned} S_{code} = Schema_{comp} \cdot \Big(&0.35Acc_{seq} + \\ &0.25Acc_{param} + 0.30C_p + 0.10Res_{valid}\Big). \end{aligned} \tag{16}$$

Table 10: **Summary of Notations and Definitions.**

Symbol	Definition	Description
Problem Formulation		
I	Natural Language Intent	The initial user query describing the experimental objective.
$\mathcal{E}$	Environmental Context	The set of available lab hardware, reagents, and physical conditions.
$\mathcal{P}$	Protocol Space	The universal set of all possible protocol sequences.
P^*	Optimal Protocol	The target executable sequence that maximizes scientific validity and physical compliance.
$\mathcal{K}_s$	Scientific Verification	Function checking experimental logic constraints (enforced by the **Scientific Reflector**).
$\mathcal{K}_p$	Physical Verification	Function checking hardware constraints (enforced by the **Rule Engine** Ω).
$\mathbb{I}[\cdot]$	Indicator Function	Returns 1 if the logical condition inside holds, 0 otherwise.
Cognitive Memory & Grounding		
$\mathcal{M}$	Cognitive Memory	The hierarchical memory system $\langle \mathcal{M}_{work}, \mathcal{M}_{episodic}, \mathcal{M}_{long} \rangle$.
Φ	Projection Function	Maps high-dimensional artifacts to symbolic pointers (Semantic Symbol Grounding).
Ψ	Resolution Function	Grounds symbolic pointers back to executable physical parameters ($a_t^{exec} \leftarrow \Psi(a_t)$).
$C_{context}$	Compressed Context	The lightweight context representation after applying Φ.
τ_t	Execution Trajectory	The history of action-observation pairs up to time step t.
Neuro-Symbolic Core (FSM)		
$\mathcal{S}$	State Space	The set of discrete cognitive states (e.g., DESIGN, VERIFY, RECTIFY).
s_t	Current State	The active state at time t, determining the valid toolset $\mathcal{A}_{valid}$.
σ	Context Signal Vector	A boolean vector $[\sigma_{know}, \sigma_{draft}, \sigma_{code}, \sigma_{sci}, \sigma_{phys}]$ driving deterministic state transitions.
Δ	Transition Function	The deterministic mapping from signals to DVR phases: $\Delta : \sigma \times \mathcal{S} \rightarrow \mathcal{S}$.
π_θ	Neural Policy	The LLM-based planner distribution over actions.
$\mathcal{A}$	Action Space	The complete set of atomic tools available to the agent.
Ω	Hardware Registry	Structured database of device limits (e.g., max RPM, temperature) defining $\mathcal{K}_p$.

D.3 System Performance Metrics

To assess operational viability, we track: **Efficiency:** Execution Time (T), Total Steps (N_{step}), and Token Consumption (N_{token}). **Success Rate ($Succ.$):** A task is deemed successful only if it meets strict domain validity criteria ($S_{code} \geq \tau$ and $C_p = 1.0$):

$$Succ. = \frac{\sum_{i=1}^{N} \mathbb{I}(\text{Task}_i \text{ is Valid})}{N_{total}}. \tag{17}$$

D.4 Statistical Analysis & Metric Verification

To ensure the rigor of our evaluation, we conducted two types of analyses: (1) Cross-model verification to assess the objectivity of our LLM-as-a-Judge metric, and (2) Paired t-tests to determine the statistical significance of our performance gains.

D.4.1 Metric Reliability (Cross-Judge)

To rule out self-preference bias in the Gemini-based judge (C_s), we re-evaluated the ablation dataset using GPT-4o Islam & Moushi (2025) and Kimi-k2-Instruct Team et al. (2025). As shown in Table 8,

Table 11: **The complete toolset definition for BioProAgent.** Tools are clustered by their function in the **Design-Verify-Rectify** loop. Note that the verification tools explicitly implement the $\mathcal{K}_s$ and $\mathcal{K}_p$ functions to drive FSM state transitions.

Tool Name	Parameters	Function Description
Phase I: Design (Generation & Alignment)		
`generate_scientific_draft`	`query`, `exp_info`, `knowledge`	Generates a natural language experimental protocol draft based on user intent.
`align_draft_to_automation`	`draft`, `exp_info`	Maps the approved draft to specific hardware operations (Intermediate step).
`generate_machine_code`	`aligned_protocol`, `suggestion`	Synthesizes executable JSON machine code. Accepts suggestions from the `Rectify` phase.
Phase II: Verify (Hierarchical Validation)		
`reflect_on_protocol`	`protocol_text`, `query`	(**Scientific Verifier** $\mathcal{K}_s$) Reviews the draft for logic/controls via CoT. Output determines the signal σ_{sci}.
`validate_machine_code`	`exp_flow_json`	(**Physical Rule Engine** $\mathcal{K}_p$) Checks JSON against the registry Ω for physical constraints. Output determines the signal σ_{phys}.
Phase III: Rectify (Self-Correction)		
`modify_protocol`	`protocol`, `request`	(**Logic Rectification**) Human-in-the-loop or auto-revision of the draft based on $\mathcal{K}_s$ feedback.
`fix_machine_code`	`machine_code`, `errors`	(**Physical Rectification**) Autonomously repairs parameters based on error logs from $\mathcal{K}_p$.
Auxiliary & Context		
`clarify_experiment_scope`	`query`, `doc_content`	Resolves ambiguities in the initial request before planning begins.
`retrieve_knowledge`	`query`, `keywords`	Fetches RAG-based literature/protocols to populate the context.
`ask_user_confirmation`	`question`	Requests explicit approval before critical execution steps.

the scoring trends across different reasoning engines are highly correlated ($r \approx 0.91$), confirming that the C_s metric objectively reflects protocol quality. GPT-4o exhibits a near-perfect correlation with Gemini ($r = 0.91$), validating the reliability of our primary metric. All three judges consistently rank BioProAgent-Full as the top performer and identify the removal of the FSM (w/o FSM) as the most detrimental ablation. Notably, GPT-4o assigns significantly lower scores to the w/o FSM variant (0.105) compared to Gemini (0.262), suggesting that advanced models penalize the lack of structured planning even more severely.

D.4.2 Statistical Significance (P-values)

We performed paired sample t-tests on the Scientific Validity scores (C_s) for the entire Subset A ($N = 160$). Table 9 reports the results for both the main baselines and the ablation variants.

E Benchmark Details

Subset A (Protocol Drafting) 160 protocols across 15 biological domains with varying query richness to test intent disambiguation and scientific reasoning; **Subset B (Code Generation)** 41 syn-

Table 12: Overview of the **Extended BioProBench**.

Benchmark Subset	Scale & Granularity	Evaluation Focus	Key Metrics
A: Scientific Drafting (Cross-domain)	160 Protocols (approx. 3k tokens/doc)	*Intent Reasoning* Knowledge retrieval	• Intent Alignment • Scientific Logic Score
B: Automation Conversion (Syn-Bio Focus)	41 Sub-Experiments (22 Device APIs)	*Local Code Mapping* Schema compliance	• Parameter Accuracy • API Validity
C: Long-Horizon Exec. (Full Pipeline)	**9 Protocols** **(547 Total Steps)**	*Global Orchestration* **State Persistence (Avg. 60.8 steps)**	• **Global Success Rate** • ID Consistency
D: Error Correction (Robustness)	30 Injection Cases (Physical/Logic Errors)	*Deterministic Robustness* Self-healing capabilities	• Detection Recall • Fix Success Rate

thetic biology sub-experiments with paired natural language drafts and ground-truth machine codes to evaluate hardware schema compliance and parameter accuracy; **Subset C (Long-Horizon)** tests 9 end-to-end long protocols, featuring up to 71 major steps, designed to test the limits of state persistence and context management; **Subset D (Error Correction)** measures 30 code snippets with injected physical/logic errors to evaluate the system's deterministic self-healing capability. We enriched the environment with API schemas for 22 specialized synthetic biology devices and a comprehensive library of consumable identifiers.

Table 13: **Complexity Statistics of the 9 Long-Horizon Tasks in Subset C.** Unlike standard benchmarks with short trajectories, these tasks involve up to 238 atomic steps and 169 unique consumables. A failure at any step leads to task failure.

Task ID	Device Interaction Nodes	Atomic Operation Steps	Involved Consumables
Sample 1	15	55	31
Sample 2	4	24	6
Sample 3	8	16	17
Sample 4	17	35	26
Sample 5	18	55	42
Sample 6	19	27	31
Sample 7	**71**	**238**	**130**
Sample 8	36	81	169
Sample 9	8	16	38
Total	**196**	**547**	**490**
Average	**21.8**	**60.8**	**54.4**

E.1 Statistical Significance of Subset C

While Subset C contains 9 high-level tasks, we argue that the sample size should be interpreted through the lens of trajectory complexity rather than task count. As detailed in Table 13, these are not single-turn queries but deep sequential decision processes. The tasks comprise a total of 547 atomic operation steps (avg. 60.8, max 238 per task). Executing Sample 7 requires managing 130 distinct consumables and 71 device interactions without a single hallucination. In wet-lab protocols, steps are strictly coupled. A deviation at Step 10 invalidates the result at Step 238.

Therefore, BioProAgent's 100% success rate in this subset represents 547 consecutive correct decisions in a grounded physical environment. If we assume a baseline agent has even a 95% step-wise accuracy, the probability of successfully completing the longest task (Sample 7, 238 steps) is negligible ($0.95^{238} \approx 5 \times 10^{-6}$). Our result demonstrates that the FSM-based architecture effectively prevents the compound error explosion typical in probabilistic LLMs.

E.2 SUBSET D: ERROR INJECTION TAXONOMY

To further investigate the robustness of BioProAgent, we present a granular breakdown of performance metrics across three specific error categories in Subset D (Error Correction). Table 14 details the agent's behavior when correcting Resource IDs, Logical Connections, and Physical Constraints.

As shown in Table 14, BioProAgent demonstrates exceptionally high *Physical Compliance* ($C_p \approx 0.95$) and *Embedding Recall* (0.98) for resource hallucinations. This indicates that while the agent correctly identifies the semantic intent and enforces safety limits, strictly formatting the JSON schema during complex repairs remains a non-trivial challenge. Crucially, the high C_p scores confirm that the neuro-symbolic FSM successfully prevents dangerous instructions even when code syntax is imperfect.

Table 14: Granular Performance Breakdown by Error Type (Subset D). While syntactic perfection remains challenging in complex repairs, BioProAgent maintains high C_p, reinforcing its role as a safety interlock.

Error Type	**Counts**	C_p	**Emb. Recall**	ACC_{seq}
Resource Grounding	10	**0.945**	**0.980**	**0.494**
Safety Alignment	10	0.890	0.955	0.430
Causal Rectification	10	0.825	0.888	0.469

F CASE STUDIES & LOGS

F.1 DETAILED EXECUTION TRACES (SUBSET C)

To provide granular insight into the "Efficiency Gap" visualized in Figure 5, we present raw execution logs from a representative long-horizon task (Task ID: `Metabolomics-Sample-Prep`). Table 15 contrasts the decision-making trajectory of BioProAgent against AutoGPT. While BioProAgent leverages its deterministic FSM backbone ($s_{t+1} = \Delta(\sigma_t)$) to strictly enforce the DESIGN-VERIFY-RECTIFY cadence, AutoGPT succumbs to a "semantic drift loop," repeatedly querying redundant information without converging on an executable plan.

F.2 THE ERROR CORRECTION GALLERY (SUBSET D)

Table 16 details specific instances of **autonomous rectification** captured during the ablation study. These examples illustrate the three primary categories of hard constraints enforced by our **Physical Rule Engine** ($\mathcal{K}_p$), demonstrating how the FSM deterministically intercepts hazardous instructions before execution.

F.3 TAXONOMY OF BASELINE FAILURES

Expanding on the trace analysis in Figure 5, we categorize the failure modes of baseline agents into two distinct pathologies based on the execution logs from Subset C:

Context Paralysis (Dominant in ReAct). As observed in 70% of ReAct failures (visualized in Figure 5, Middle), the agent correctly retrieves the schema but fails to synthesize it into code. The logs show abrupt termination with empty outputs. This confirms that without **Symbol Grounding** (Φ), the raw HTML/JSON payloads from API docs quickly saturate the LLM's working memory, leading to a cognitive freeze.

Info-Seeking Loop (Dominant in AutoGPT). In 90% of AutoGPT failures (visualized in Figure 5, Bottom), the agent enters a recursive retrieval loop (e.g., querying buffer composition $\rightarrow$ "buffer pH" $\rightarrow$ "buffer supplier"). Unlike BioProAgent's deterministic FSM, which mandates a rigid transition from `DESIGN_DRAFT` to `DESIGN_CODE`, AutoGPT lacks a structural stopping condition, optimizing for *information completeness* rather than *task completion*.

Table 15: **Side-by-Side Execution Log Comparison.** BioProAgent utilizes FSM states to progress deterministically, completing the task in 8 steps. In contrast, AutoGPT enters a semantic drift loop, exhausting the context window (Step 45) without producing valid code.

Step	BioProAgent (Ours)	AutoGPT (Baseline)
1	**[State: CLARIFY_INTENT]** *Trigger:* $\sigma_{ambiguous}$ *Action:* `clarify_experiment_scope(query=...)` *Output:* "Sample Volume: 50uL, Kit ID: 10038"	**[Thought]** I need to find the protocol for metabolomics. *Action:* `retrieve_knowledge(query="metabolomics prep")`
2	**[State: DESIGN_DRAFT]** *Trigger:* $\sigma_{know} \wedge \neg\sigma_{draft}$ *Action:* `generate_scientific_draft(...)` *Grounding* (Φ)*:* Mapped "Kit_A" $\rightarrow$ `ID:10038`	**[Thought]** I found some docs. Now I need specific reagent list. *Action:* `retrieve_knowledge(query="reagent list")`
3	**[State: DESIGN_CODE]** *Trigger:* $\sigma_{approved} \wedge \neg\sigma_{code}$ *Action:* `generate_machine_code(...)`	**[Thought]** I need to check the temperature settings. *Action:* `retrieve_knowledge(query="temperature settings")`
...	*(Step 4-7: Validation $\mathcal{K}_s$, $\mathcal{K}_p$ passed)*	*(Repeated retrieval loops for 30+ steps...)*
8 / 45	**[State: SUCCESS]** *Trigger:* $\sigma_{code} \wedge \sigma^{+}_{phys}$ *Result:* Valid JSON Code committed. **Task Success (Total Tokens: 10,329)**	**[Error] Max Context Exceeded.** *Result:* Timeout / Fail. **Task Failure (Total Tokens: 59,278)**

Table 16: **Gallery of Autonomous Corrections.** The table compares the raw design violation (left), the specific safety interlock triggered by the Rule Engine (center), and the rectified code after FSM-guided feedback (right).

Constraint	Design Violation (s_{design})	Interlock Trigger ($\neg\mathcal{K}_p$)	FSM Rectification ($s_{rectify}$)
Safety	`op.centrifuge(speed=25,000g, duration=15min)`	**[HALT]** Speed 25,000g exceeds rotor limit (Max: 15,000g).	`op.centrifuge(speed=15,000g, duration=15min)` *# Clamped to safe limit*
Resource	`op.transfer(source=new_buf, vol=50)`	**[HALT]** Resource ID `new_buf` not found in Registry Ω.	`op.transfer(source=plate_3, vol=50)` *# Mapped to valid ID*

Logic	`op.seal_plate()`
	`op.add_reagent(vol=10)`
		[WARN] Cannot `add_reagent` to a sealed container (State Conflict).	`op.add_reagent(vol=10)`
			`op.seal_plate()`
			# Sequence reordered

G Hardware Registry and Action Space Schema

To support reproducibility and the reconstruction of the **Physical Rule Engine** ($\mathcal{K}_p$), we formally define the device action space. The Hardware Registry (Ω) contains 22 specific instruments. Every action a_t generated by the agent must strictly conform to the parameterized schemas defined below.

G.1 Device Categories and Functions

The registry organizes hardware into four functional categories to enable end-to-end wet-lab automation:

- **Liquid Handling:** Automated pipetting workstations, acoustic liquid handlers, and dispensers.
- **Thermal Control:** PCR thermal cyclers, incubators, and plate sealers.
- **Separation & Processing:** Centrifuges, shakers, and magnetic separators.
- **Analysis:** Multi-mode plate readers, qPCR systems, and electrophoresis imagers.

Table 17: **Representative Action Schemas from the Hardware Registry (Ω).** The Physical Rule Engine ($\mathcal{K}_p$) validates generated JSON code against these parameter constraints and physical interlocks.

Device Category	Action Primitive	Parameters (Type: Constraint)	Physical Rules (Examples)
Centrifugation	`Centrifuge`	• `speed_g` (int: 500-15000) • `time` (duration: HH:MM:SS) • `temp_C` (int: 4-40) • `brake` (enum: off, slow, fast)	1. Speed must not exceed rotor limit ($V \leq V_{max}$). 2. Temperature requires pre-cooling if $T < 25°$C.
Thermal Control	`Incubate`	• `temp_C` (int: 4-99) • `shake_rpm` (int: 0-2000) • `duration` (time)	1. Shaking prohibited if plate is unsealed. 2. Max temp depends on plate material (PS vs. PP).
Liquid Handling	`Transfer`	• `source` (ID string) • `dest` (ID string) • `volume_uL` (float: 0.5-1000) • `tip_type` (enum: p20, p300, p1000)	1. Source volume must be $>$ aspiration volume + dead volume. 2. Tip type must match volume range.
PCR Cycling	`Thermocycle`	• `lid_temp` (int: 95-105) • `stages` (list of [temp, time, cycles])	1. Lid temp must be $>$ reaction temp to prevent condensation.

G.2 Action Primitives and Parameter Constraints

Table 17 specifies the schemas for representative action primitives. All constraints (e.g., integer ranges, enumerations) are strictly enforced by the **Physical Rule Engine ($\mathcal{K}_p$)** during the `VERIFY` phase.

G.3 Full Registry Availability

Due to space constraints, Table 17 presents a subset of the 22 devices. The complete JSON schema definitions, including configurations for Gas Evaporators, Colony Pickers, and Plate Washers, are provided in the supplementary material.

H System Prompts

To support reproducibility, we provide the core system prompts used in BioProAgent. Variable placeholders are denoted by brackets (e.g., {`user_input`}).

H.1 FSM-Driven Planner Prompt

The planner prompt illustrates how the Finite State Machine (FSM) explicitly governs the agent's trajectory. Note the `State->Action Mapping` section, which enforces the **Priority Decision Matrix** described in Section 3.2 (Table 1), ensuring that the agent strictly adheres to the **Design-Verify-Rectify** workflow.

```
You are BioProAgent, a bio-experiment planning agent. Your task is to convert user requests
     into executable tool call sequences.

## Scientific Principles
- Reproducibility: Steps are clear and reproducible.
- Control Principle: Negative control (NTC), Positive control (PTC), Blank control.
- Variable Control: Change only one variable at a time.

## Current State: {current_state}
## Suggested Actions: {state_guidance}

## User Request
{user_input}

## Experiment Context
{experiment_context}

## Executed Steps
{execution_summary}

## Available Tools
{tools_description}
```

```
## Output Format Requirements (Strictly Follow)
Output must be a JSON array.
Example: [{"tool_name": "retrieve_knowledge", "args": {...}}]

### State -> Action Mapping (Hard Constraints)
- CLARIFY_INTENT -> [clarify_experiment_scope]
- DESIGN_DRAFT   -> [retrieve_knowledge, generate_scientific_draft]
- VERIFY_DRAFT   -> [reflect_on_protocol]
- RECTIFY_DRAFT  -> [modify_protocol, reflect_on_protocol]
- DESIGN_CODE    -> [align_draft_to_automation, generate_machine_code]
- RECTIFY_CODE   -> [fix_machine_code, validate_machine_code]
- SUCCESS        -> [add_memory]

## Important Instructions
- Follow scientific principles strictly.
- Current state is **{current_state}**
- Suggested actions: {state_guidance}
- The FSM state determines your valid action space. Do not hallucinate tools outside the
    mapping.
```

Listing 1: The Neuro-Symbolic Planner Prompt

H.2 NEURO-SYMBOLIC ALIGNMENT PROMPT

This prompt operationalizes the **Semantic Symbol Grounding** (Φ) phase. It maps natural language protocols to the hardware registry by strictly categorizing steps into `[AUTO]`, `[EXTERNAL]`, and `[MANUAL]`, thereby preventing context overflow.

```
You are a biological experiment automation expert.
Task: Convert the [Reference Info] into a standardized executable process.

## Step Tagging Rules (Critical!)
Every step must be tagged with one of the following:

| Tag | Meaning | Usage |
|-----|---------|-------|
| **[AUTO]** | Executable by platform | Use available devices below |
| **[EXTERNAL]** | Requires external device | Lab has device, but not integrated |
| **[MANUAL]** | Requires human operation | Decision making or special ops |

**Important**: [EXTERNAL] and [MANUAL] are NOT failures. They are normal parts of complex
    biological workflows.

## Available Automation Devices (Summary)
- Pipetting Workstation: Liquid transfer (0.5uL-1mL), mixing, temp control
- PCR Machine: Temp cycling (4-99C)
- Sealer/Peeler: Plate sealing
- Centrifuge: Separation (500-15000g)
- Incubator: Constant temp (20-60C)
- Plate Reader: Absorbance/Fluorescence

## Output Format Specification
1. [AUTO] Pipetting Workstation - Setup PCR Reaction
   1.1 [AUTO] Pipetting Workstation - Add Template DNA (Vol=2uL)
     - Reagent: Template DNA
     - Consumable: PCR Plate
   1.2 [AUTO] Pipetting Workstation - Add Master Mix
2. [AUTO] Sealer - Seal Plate (Temp=170C)
3. [AUTO] PCR Machine - Amplification
   - Program: 95C 5min -> [95C 30s, 58C 30s, 72C 1min] x35
4. [EXTERNAL] Gel Electrophoresis - Check Products
   - Note: Requires external imaging system
5. [MANUAL] Human Decision - Verify Band Size
   - Criteria: Target band at ~500bp

## Input Information
User Requirements: {exp_info}
Reference Protocol: {protocol}
```

Listing 2: Protocol Alignment and Device Grounding Prompt

H.3 BASELINE SOP INJECTION

To ensure fair comparison, baselines (ReAct, Direct LLM) were injected with a Standard Operating Procedure (SOP) prompt. This minimizes naive formatting errors and ensures that performance gaps are attributable to reasoning architecture rather than prompt engineering.

```
## Laboratory Standard Operating Procedures (SOP)
You are a Senior Laboratory Automation Engineer. To ensure success, you MUST follow these
     protocols:

1. **Tool Priority**: Do NOT hallucinate JSON formats.
   - Use `generate_machine_code` to create the final JSON.
   - This tool knows the correct Schema (nodes, resourceId, connections).

2. **Validation Loop**:
   - After generating code, you MUST use `validate_machine_code`.
   - If validation fails (red light), use `fix_machine_code` or regenerate.
   - Do NOT output the final answer until validation passes.

3. **Data Handling**:
   - Use data references (e.g., $draft, $machine_code) to pass information between tools to
        avoid context overflow.
```

Listing 3: Standard Operating Procedure (SOP) for Baselines