Review

Not peer-reviewed version

Large Language Models for Reinforcement Learning: A Survey of Intervention Operators and Optimization Effects

Kourosh Shahnazari *, Seyed Moein Ayyoubzadeh , Mohammadali Keshtparvar

Posted Date: 3 March 2026

doi: 10.20944/preprints202603.0229.v1

Keywords: large language models; reinforcement learning; planning priors; reward shaping; verification; credit assignment

 doi:10.20944/preprints202603.0229.v1

Review

Large Language Models for Reinforcement Learning: A Survey of Intervention Operators and Optimization Effects

Kourosh Shahnazari [1,*,†], Seyed Moein Ayyoubzadeh [1,†] and Mohammadali Keshtparvar [2]

[1] Sharif University of Technology, Tehran, Iran
[2] Amirkabir University of Technology, Tehran, Iran
* Correspondence: kourosh@null.net
† These authors contributed equally.

Abstract

Reinforcement learning is increasingly deployed in domains where reward feedback is sparse, delayed, and entangled with long-horizon constraints, making reliable credit assignment difficult. A central development in recent work is the insertion of large language model modules directly into the reinforcement learning loop, not as peripheral interfaces but as components that alter trajectory generation and supervision. In these systems, language modules provide planning priors, structured reward shaping, process verification, synthetic world traces, and tool-memory context that reconfigure optimization at trajectory level. This survey develops a mechanism-first synthesis of that shift. We formalize intervention operators for planning, reward and verifier channels, world construction, and tool-memory mediation; analyze how each operator changes update targets, bias pathways, and stability conditions; and organize the field into a unified taxonomy grounded in optimization effects rather than model branding. We then examine evaluation practice across embodied control, web interaction, games, continuous control, and multi-agent settings, highlighting reproducibility gaps and protocol confounds. Finally, we synthesize recurring failure modes and propose a concrete research agenda on calibration, module authority arbitration, uncertainty-aware simulation, and benchmark design. The resulting perspective positions LLM-in-the-loop reinforcement learning as a systems and optimization discipline centered on trustworthy credit assignment under heterogeneous supervision.

Keywords: large language models; reinforcement learning; planning priors; reward shaping; verification; credit assignment

1. Introduction

1.1. Motivation

Reinforcement learning has produced strong results in narrow, well-shaped environments, but its practical limits become visible when decision-making moves into open-ended settings with delayed feedback, combinatorial action structure, and partial observability. In robotics, a manipulation sequence may succeed or fail only after a long chain of intermediate contacts and safety-sensitive motions. In web interaction, success often requires stateful navigation through dynamically changing interfaces where many early actions appear locally plausible but are globally harmful. In strategic games and text-rich interactive domains, reward sparsity and horizon length make gradient signals weak precisely where policy differentiation is most needed. Under these conditions, optimization fails less because of policy parameterization and more because supervision arrives in forms that are too coarse, too delayed, or too ambiguous to support stable credit allocation.

This weakness is magnified when the environment objective is underspecified. Reward design in realistic tasks is frequently an engineering bottleneck: hand-crafted reward terms can be fragile, easy to exploit, or expensive to maintain under distribution shift. Even when reward is available, the same

 doi:10.20944/preprints202603.0229.v1

scalar return can correspond to many qualitatively different trajectories, some robust and some brittle. Standard temporal-difference updates do not automatically encode process quality, causal structure, or semantic constraints. As a result, training often converges toward shortcut behaviors that satisfy proxy feedback while degrading out-of-distribution performance.

Recent systems increasingly address these failures by embedding language modules inside the control loop itself. Instead of treating a language model as an outer wrapper that only generates high-level instructions, these methods route language outputs into the trajectory-generation and update pipeline: subgoal proposals shape exploration, reward and verifier channels densify supervision, synthetic rollouts modify replay, and tool-memory interfaces alter effective state. This is a structural intervention, not a cosmetic one. Once multiple channels participate in update construction, the learning system no longer optimizes solely against raw environment reward; it optimizes against a composite supervision process whose reliability depends on module calibration and arbitration policy [1–6].

A second motivation comes from horizon management. Many long-horizon tasks are difficult because the agent must discover useful intermediate structure before gradients become informative. LLM-driven decomposition can expose semantically coherent latent plans, reducing effective horizon and focusing exploration on causally relevant regions of state-action space. The effect is especially visible in hierarchical control settings where high-level plans, option proposals, or code-like sketches provide scaffolds that low-level RL can refine through grounded interaction [7–10].

A third motivation is state construction under partial observability. In many interactive tasks, useful state is distributed across logs, tools, documents, and previous reasoning traces. Tool-memory modules can integrate these sources into a persistent context that changes both action selection and the meaning of subsequent rewards. This creates opportunities for better long-term consistency but also introduces new failure surfaces when stale or erroneous context propagates across decisions [11–14].

Viewed at system level, these developments represent a shift from monolithic policy learning toward modular optimization pipelines. In monolithic designs, supervision quality is mostly determined by reward design and value estimation. In modular designs, supervision quality is co-determined by planner priors, verifier calibration, simulation fidelity, and state-construction reliability. The resulting system can be far more capable, but only if these channels are coordinated. This is why many recent gains are accompanied by new classes of failure that do not appear in classical RL benchmarks, including verifier-target mismatch, synthetic replay drift, and tool-context contamination.

This survey argues that the right conceptual unit is not the model, but the intervention pathway. A planner is useful not because it is linguistic, but because it changes which trajectories are explored. A verifier is useful not because it can critique text, but because it changes which updates are trusted. A world module is useful not because it can narrate dynamics, but because it changes replay composition and planning rollouts. Framing the field this way allows architectural comparisons across domains and avoids category confusion between language capability and optimization function.

The same pathway view also clarifies why optimistic claims often fail to transfer across task families. A method that improves robotic manipulation through decomposition may underperform in web agents if tool contracts are unstable. A verifier channel that improves textual process quality may destabilize low-level control if score semantics are weakly correlated with environment return. These outcomes are not contradictions; they reflect different pathway sensitivities. Interpreting results through intervention pathways therefore improves external validity by separating what is module-specific from what is mechanism-general.

A final motivation is methodological maturity. As the literature grows, architectural names proliferate faster than shared design principles. Without a unifying lens, neighboring methods are compared as if they were fundamentally different even when they manipulate the same update pathway. Conversely, methods with similar reported metrics may rely on incompatible intervention contracts. The field now benefits more from synthesis of pathway-level assumptions and authority policies than from additional ad hoc module variants.

 doi:10.20944/preprints202603.0229.v1

1.2. Thesis and Scope Boundary

The thesis of this survey is that LLM-enhanced reinforcement learning is best understood as trajectory-level optimization under heterogeneous supervision. The relevant question is not whether language models are powerful in general, but how specific language-mediated operators intervene in sampling, weighting, and update construction inside the RL loop. This perspective unifies apparently diverse systems and clarifies why similar gains and failure modes appear across robotics, web agents, game-playing, and offline regimes.

Scope is directional. We survey methods where language modules improve reinforcement learning systems by influencing policy learning, planning, reward shaping, verification, simulation, or state mediation in environment-grounded tasks. Work centered on using RL to improve language models themselves is adjacent context but not the object of analysis here. This boundary matters because the optimization target differs fundamentally: in-scope work aims to improve decision quality and learning dynamics of RL agents; out-of-scope work primarily targets linguistic behavior of the language model.

This directional criterion also avoids conceptual conflation. A method can involve both RL and LLMs while still being irrelevant to RL-system improvement if the environment-grounded control objective is absent. Conversely, a method can remain in scope even when language modules are not the final action policy, as long as they alter trajectory priors, supervision density, or state construction in ways that measurably improve RL performance.

This boundary is especially important for credit-assignment analysis. When the optimized object is an RL agent interacting with an environment, language channels act as supervision modifiers whose quality can be evaluated against grounded returns. When the optimized object is the language model itself, those same channels serve a different objective and different failure modes dominate. Conflating these settings would blur causal interpretation of reported gains. By keeping scope directional, we maintain a coherent optimization unit throughout the survey.

1.3. Search Strategy and Selection Criteria

To provide representative coverage of this rapidly evolving area, we conducted an iterative literature search centered on arXiv and Google Scholar, complemented by venue-focused screening of major AI conference proceedings, including NeurIPS, ICML, ICLR, ACL, and AAAI, and by targeted checks of relevant journal publications in machine learning and artificial intelligence. The primary time window was 2022–2026, which captures the period in which large language model modules became explicit components of reinforcement learning pipelines. Earlier work was incorporated when conceptually necessary to ground operator-level formulations, especially for planning priors, reward shaping theory, verifier-based supervision, model-based rollouts, and replay/data-distribution effects.

The query process used combinations and variants of terms such as LLM in reinforcement learning, LLM-augmented RL, language models for planning, LLM reward shaping, process supervision in RL, LLM world model, LLM tool use RL, LLM policy guidance, and LLM-based trajectory scoring. Candidate papers were then assessed for mechanism relevance rather than branding or benchmark popularity. Inclusion required that the language model intervene directly in the reinforcement learning optimization loop, for example through planner guidance, reward or verifier channels, world-model construction, replay or trajectory reweighting, or state mediation through memory and tools, and that the work provide empirical evidence, algorithmic novelty, or both.

We excluded studies where RL was used primarily to optimize the language model itself (e.g., RLHF-style objectives), pure RL work without language-model intervention, and agentic systems that used language models for orchestration without a clear coupling to optimization targets or update pathways. To the best of our knowledge, this process yields representative coverage rather than exhaustive completeness. The final corpus comprises 125 selected works, which we map into the mechanism-level taxonomy introduced in this survey to support pathway-specific comparison and synthesis.

 doi:10.20944/preprints202603.0229.v1

1.4. Contributions of This Survey

This survey makes five contributions. First, it formalizes a unifying framework in which language modules are intervention operators that act on trajectory generation and update pathways. Second, it proposes a mechanism-first taxonomy organized by optimization effects: trajectory prior shaping, supervision restructuring, world construction, search hybrids, data augmentation, and tool-memory mediation. Third, it develops a credit-assignment analysis that distinguishes beneficial densification from bias injection and emphasizes the role of module authority in controlling that trade-off. Fourth, it synthesizes evaluation practice across domain families and identifies protocol-level confounds that limit comparability. Fifth, it consolidates recurring failure modes and articulates a concrete research agenda aimed at calibrated verification, uncertainty-aware simulation, authority arbitration, and reproducible benchmark design.

1.5. Organization of the Paper

Section 2 introduces the minimal RL and LLM preliminaries needed for subsequent analysis. Section 3 presents the unifying framework, formal intervention operators, and core equations for mixed supervision. Section 4 develops the mechanism-first taxonomy with category-level synthesis, architectural patterns, and representative evidence. Section 5 discusses evaluation ecosystems and reporting standards. Section 6 analyzes system-level failure modes and trade-offs. Section 7 outlines ten open research problems. Section 8 concludes with a synthesis of implications for dependable long-horizon RL.

2. Background and Preliminaries

2.1. RL Objective and Trajectory View

We consider a Markov decision process $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$ with policy $\pi_\theta(a \mid s)$. The optimization objective is

$$J(\pi) = \mathbb{E}_\tau \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t) \right], \tag{1}$$

where τ is a trajectory sampled under the policy and environment dynamics. Equation (1) emphasizes that learning quality depends jointly on trajectory distribution and reward informativeness. In long-horizon tasks, two policies can produce similar short-term rewards while diverging sharply in delayed outcomes; thus trajectory structure, not only instantaneous action accuracy, governs final performance.

A trajectory is written as

$$\tau = (s_0, a_0, r_0, \ldots, s_T). \tag{2}$$

Equation (2) is central because every LLM intervention considered in this survey ultimately modifies one of its components: state representation, action proposal, reward channel, or replay composition. Treating LLM modules as trajectory operators clarifies when gains derive from true control improvement versus superficial proxy optimization.

For policy-gradient methods, a compact expression is

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}\left[\sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)\, \hat{A}_t \right]. \tag{3}$$

Equation (3) highlights where credit assignment enters: the advantage estimate $\hat{A}_t$ determines how strongly each action affects updates. Sparse or delayed rewards make $\hat{A}_t$ noisy and high-variance, which motivates auxiliary supervision channels that can densify learning signals. However, any auxiliary channel can also introduce systematic bias if it is misaligned with environment outcomes.

For value-based reasoning, the Bellman relation provides the recursive backbone:

$$V^\pi(s) = \mathbb{E}_{a \sim \pi,\, s' \sim P}\left[r(s, a) + \gamma V^\pi(s') \right]. \tag{4}$$

 doi:10.20944/preprints202603.0229.v1

Equation (4) clarifies why trajectory quality and representation quality are inseparable: errors in state construction or reward interpretation propagate through recursive targets and can amplify over horizon.

2.2. Credit Assignment Pathways

In conventional RL, supervision primarily flows from environment reward through returns, value targets, or temporal-difference errors. This pathway is grounded but often weak in open-ended tasks. Reward shaping and auxiliary losses are therefore common, but their reliability depends on how faithfully they track task success over long horizons.

The key challenge is temporal and structural ambiguity. Long delays obscure causal contribution of early actions; dense interaction logs may include many irrelevant steps; and partial observability means that critical state information may never be directly encoded in raw observations. Credit assignment therefore becomes a systems problem: what information channels are trusted, when they are trusted, and how their influence is weighted during updates.

Offline RL and model-based RL expose complementary versions of this issue. Offline methods inherit dataset support constraints; new supervision channels can help by relabeling or augmenting data, but can also worsen extrapolation error if synthetic samples are unrealistic. Model-based methods can improve sample efficiency through imagined rollouts, but model bias can dominate when simulated dynamics drift from real transitions [15–18].

From an estimation viewpoint, each additional supervision pathway perturbs both bias and variance. Dense shaping often reduces variance by supplying frequent learning signals, yet it can increase bias if the shaping function tracks stylistic regularities rather than causal progress. Verifier gating can reduce variance by rejecting noisy trajectories, but can increase bias when calibrated confidence diverges from true task utility. Offline augmentation can reduce variance in low-support regions, but can amplify bias if synthetic traces systematically miss environment constraints. This recurring bias-variance exchange is a central thread across all taxonomy categories discussed later.

2.3. What LLM Modules Add in Decision Settings

Language modules contribute three capabilities that are directly relevant to RL optimization. First, they provide structured priors over latent plans, subgoals, and action abstractions, which can reduce search complexity. Second, they can transform weak scalar feedback into denser process supervision by generating reward terms, critiques, or verifier scores. Third, they can mediate state by integrating tool outputs and memory traces into contextual summaries used for action selection and update filtering.

These capabilities are useful because they target the same bottlenecks that cause instability in long-horizon RL: sparse supervision, weak decomposition, and incomplete state. Yet usefulness is conditional. Any module output enters learning as a model-dependent signal with uncertainty and potential bias. High performance therefore requires explicit arbitration between grounded environment feedback and language-derived channels rather than unconditional trust in generated content [19–22].

From this perspective, "LLM for RL" is not a single method class but a family of control interventions with different statistical properties. Planner channels primarily affect exploration and support. Reward and verifier channels primarily affect target construction. Simulation channels primarily affect data distribution. Tool-memory channels primarily affect observability. Each pathway introduces a distinct bias-variance profile, and practical systems must tune these profiles jointly. This motivates the mechanism-first taxonomy adopted later in the paper.

Another implication is that evaluation should move beyond aggregate return reporting. Two systems can achieve similar return with very different supervision structures and thus very different robustness properties. A planner-heavy method may fail under decomposition shift, while a verifier-heavy method may fail under reward-proxy exploitation. Without pathway-specific diagnostics, these differences remain hidden until deployment.

The broader methodological consequence is that experiment design should align with mechanism claims. If a paper claims improvement through planning priors, it should report decomposition robust-

 doi:10.20944/preprints202603.0229.v1

ness and feasibility diagnostics. If it claims improvement through verifier channels, it should report calibration and anti-gaming evidence. If it claims gains through world construction, it should quantify synthetic-real divergence and uncertainty control. This alignment between claimed mechanism and reported evidence is still inconsistent in the literature, and strengthening it would materially improve interpretability and reproducibility.

The same logic motivates stronger reporting of intervention contracts. A contract specifies what a module may output, where that output enters optimization, and how influence is bounded under uncertainty. Contracts make comparisons fairer because they expose whether gains come from model quality, authority level, or hidden implementation details. In their absence, replicability suffers: apparently minor choices such as memory truncation, tool timeout handling, or verifier thresholding can dominate outcomes. A contract-centric perspective therefore complements algorithm-centric benchmarking and supports clearer cumulative progress.

Finally, preliminaries should be read as a map of supervision pathways, not as a separate background chapter detached from method design. The same policy-gradient estimator can behave very differently depending on which pathway contributes to $\hat{A}_t$ and how much authority each pathway receives. This insight motivates the next section, where module interventions are formalized as operators with explicit semantics and bounded authority.

3. Unifying Framework: LLM Modules Embedded in the RL Loop

3.1. Formal Intervention View

Let M_ϕ denote a language module with parameters ϕ and context input h_t that includes state history, tool traces, and optional memory. The module emits an intermediate variable o_t:

$$o_t = M_\phi(h_t). \tag{5}$$

The policy can then condition on augmented state and module outputs. This formalization treats language outputs as control-relevant latent variables rather than free-form text artifacts.

A planner module instantiates this idea through latent subgoal variables:

$$\pi(a_t \mid s_t, z_t), \qquad z_t = \text{Planner}_\phi(h_t). \tag{6}$$

Equation (6) captures hierarchical conditioning. The planner compresses long-horizon structure into shorter-horizon subproblems. Benefits arise when subgoals are feasible and informative; instability arises when generated plans are semantically coherent yet dynamically incompatible with low-level control.

A reward-shaping module modifies immediate training targets:

$$r'_t = r_t + \alpha\, b_t, \tag{7}$$

where b_t is language-derived shaping and α controls authority. Equation (7) can reduce variance and accelerate learning, but only if shaped terms preserve task semantics. If b_t overweights superficial process features, optimization may drift toward proxy behaviors.

One theoretical anchor is potential-based shaping: if the shaping term corresponds to a potential difference over states, it preserves optimal policies while improving learning speed. Language-derived shaping rarely satisfies such conditions by default, so shaping should be treated as an authority parameter that demands calibration, ablation, and shift stress tests rather than as a guaranteed "free" improvement.

Verifier-mediated weighting can be expressed as an expectation over reweighted trajectory loss:

$$\mathbb{E}_\tau[w(\tau)\, L(\tau)], \tag{8}$$

 doi:10.20944/preprints202603.0229.v1

with $w(\tau)$ derived from verifier scores. Equation (8) clarifies that verifier channels do not only add information; they reshape effective data distribution seen by the optimizer. This mechanism can suppress noisy trajectories, but it can also hide failure modes if the verifier is systematically overconfident.

This perspective also connects verifier weighting to off-policy importance weighting. As in importance sampling, heavy-tailed weights can destabilize updates, while aggressive clipping or normalization trades some bias for lower variance. In practice, calibrated trust and bounded weights are often more robust than hard gating, particularly when the verifier distribution lags policy drift.

Offline augmentation introduces synthetic traces:

$$\mathcal{D}' = \mathcal{D} \cup \hat{\mathcal{D}}, \tag{9}$$

where $\hat{\mathcal{D}}$ contains generated or relabeled trajectories. Equation (9) highlights the central trade-off: broader coverage versus distribution shift. Effective systems bound synthetic influence with confidence-aware filters and consistency checks against real rollouts.

Tool-memory mediation can be written as

$$m_{t+1} = U(m_t, \text{tool}(s_t), o_t), \tag{10}$$

where m_t is persistent context, $\text{tool}(s_t)$ denotes external retrieval or API observations, and o_t is module output used for memory update decisions. Equation (10) makes explicit that memory is not passive storage; it is a learned state-construction process. Errors in U can persist across many steps and alter both action selection and subsequent supervision interpretation.

Taken together, Eqs. (6)–(10) describe a family of operators that intervene at different points but couple through shared trajectories. This coupling explains why isolated ablations may underestimate risk: a planner failure can be amplified by a verifier; a memory error can distort planner context; a world-model bias can change verifier inputs. Reliable design therefore depends on joint reasoning across operators, not independent tuning.

3.2. Intervention Operators and Channel Semantics

The framework uses five operator classes. The planner operator emits latent decomposition variables that alter exploration and option selection. The reward operator emits shaping terms that modify immediate targets. The verifier operator emits process scores that gate, rank, or reweight trajectory updates. The world operator emits imagined transitions that affect planning or replay. The tool-memory operator updates contextual state from retrieval and execution traces.

These operators are compositional. A single system can use planner outputs to propose subgoals, verifier scores to select among candidate rollouts, and world-model traces to train value functions. Compositionality creates expressive power but complicates diagnosis: when performance changes, attribution must separate planner quality, verifier calibration, simulator reliability, and arbitration policy.

Channel semantics matter as much as model architecture. A verifier score interpreted as hard gating behaves differently from the same score used as soft weighting. A world-model trace used for candidate ranking differs from one used as direct replay data. Robust design therefore requires explicit contracts for each channel: what it can change, when it can change it, and how confidence affects authority.

3.3. Credit Assignment Implications

LLM modules reshape credit assignment through three mechanisms. First, they densify supervision by converting delayed outcomes into intermediate signals. Second, they alter trajectory priors through decomposition and search guidance. Third, they modify data distribution through filtering and synthetic augmentation. Each mechanism can improve sample efficiency, but each introduces model-dependent bias.

 doi:10.20944/preprints202603.0229.v1

Densification improves optimization when auxiliary signals preserve causal relevance. If shaping and verifier channels track process quality that predicts long-term return, advantage estimates become more informative and variance drops. If they track easily exploitable proxies, the same channels amplify misalignment. This explains why verifier calibration and disagreement monitoring are recurring best practices [19,20,23,24].

Trajectory-prior interventions improve horizon management when planner proposals respect environment dynamics. In hierarchical settings, language decomposition can reduce combinatorial search and accelerate curriculum progression. Yet planner authority must be bounded by feasibility checks; otherwise policy updates are trained on unattainable intent structures and convergence degrades [6,25,26].

Distribution-shaping interventions are especially sensitive. Synthetic trajectories can fill support gaps in offline or expensive environments, but unfiltered generation can create high-confidence error cascades. Practical systems now combine synthetic replay quotas, uncertainty-aware weighting, and periodic re-anchoring to real interaction return [27–29].

An underappreciated implication is that module interventions change effective objective geometry even when nominal reward is unchanged. Reweighting alters which trajectories dominate gradient estimates; shaping alters local smoothness of learning signals; decomposition alters reachable policy neighborhoods under finite compute. These effects can improve optimization conditioning, but they can also create deceptive plateaus where auxiliary channels appear to converge while grounded return stagnates. Diagnosing this gap requires channel-specific monitoring rather than aggregate reward alone.

This viewpoint suggests a practical bias-variance decomposition for modular RL updates. Planner and verifier channels often reduce estimator variance by concentrating updates on structured trajectories, yet they can increase bias when their semantics drift from environment utility. Generator channels can reduce variance through broader replay support while introducing model bias through synthetic mismatch. Tool-memory channels can reduce partial-observation variance but introduce temporal bias if stale context persists. Effective credit assignment therefore depends on balancing channel-specific bias and variance contributions, not on maximizing any single auxiliary signal.

This decomposition connects naturally to established RL theory. When shaping terms can be approximated as potential differences, they tend to preserve policy ordering while reducing temporal sparsity; when they cannot, they still may help optimization but lose invariance guarantees and require stronger grounding checks [4,30,31]. Verifier-based reweighting is similarly close to clipped importance weighting in off-policy updates: bounded weights can reduce estimator variance, but miscalibrated weights can shift effective training distribution away from environment utility [19,20,24].

Generator channels mirror the classic Dyna trade-off between sample efficiency and model bias. Imagined rollouts can accelerate value propagation, especially in low-interaction regimes, but bias grows with rollout depth and transition misspecification [15–17,32]. A consistent implication across these settings is that auxiliary channels should be treated as estimators with uncertainty budgets and explicit authority constraints, not as direct substitutes for grounded environment feedback.

For verifier-mediated updates, an off-policy analogy is useful:

$$w(\tau) = \mathrm{clip}(\rho(\tau), 0, c), \tag{11}$$

where $\rho(\tau)$ is a surrogate likelihood or quality ratio and c is a clipping constant. In LLM-enhanced RL, $\rho(\tau)$ is usually unavailable in closed form and is approximated by verifier confidence, which reinforces the need for calibration-aware clipping and periodic grounding checks [19,20,23].

3.4. Module Authority and Arbitration

A practical way to reason about safety and stability is module authority. Advisor-level modules propose candidates without direct control. Controller-level modules affect action selection or reward terms. Judge-level modules gate updates or trajectories. Simulator-level modules influence the data

 doi:10.20944/preprints202603.0229.v1

regime itself. As authority increases, potential sample-efficiency gains increase, but so does the risk of systematic error.

Arbitration policies determine whether authority is static or adaptive. Static weighting is simple but brittle under shift. Adaptive arbitration uses uncertainty, disagreement, and environment-grounded diagnostics to adjust influence online. Empirically, systems with explicit arbitration tend to degrade more gracefully when tool quality, prompt format, or domain dynamics change.

3.5. Concrete Arbitration Recipes

Two practical schedules can be implemented with minimal overhead and provide stable authority control in mixed-supervision training.

$$\alpha_t = \alpha_0 \cdot \sigma(\kappa(1 - \mathrm{ECE}_t)), \tag{12}$$

where α_0 is the maximum shaping weight, $\sigma(\cdot)$ is the logistic function, κ controls sensitivity, and ECE_t is expected calibration error measured on a rolling buffer of verifier predictions versus delayed grounded outcomes. In practice, ECE_t can be recomputed every N updates; when calibration worsens, α_t contracts automatically, limiting shaped-reward authority.

$$w(\tau) = \min\left(1, \frac{\beta}{\hat{\sigma}_{\text{verifier}}(\tau) + \epsilon}\right), \tag{13}$$

where $\hat{\sigma}_{\text{verifier}}(\tau)$ is trajectory-level uncertainty (ensemble variance or dropout variance), β is a trust budget, and ϵ avoids numerical instability. This schedule is directly usable in replay sampling or loss reweighting: uncertain trajectories still contribute but with bounded influence. Combined with periodic disagreement checks across verifier heads, this yields a simple advisor-to-judge authority schedule that is both auditable and robust under moderate shift.

In deployed pipelines, both schedules are typically computed on rolling windows: (i) update ECE_t from recent verifier predictions versus delayed grounded outcomes, (ii) update $\hat{\sigma}_{\text{verifier}}$ from ensemble variance, and (iii) apply conservative floor and ceiling constraints on α_t and $w(\tau)$ to prevent abrupt authority shifts. This workflow is lightweight enough for online adaptation and has been adopted in variants of verifier-weighted and search-guided systems [19,20,24,33].

Algorithm 1 Authority escalation with rollback triggers

Initialize authority bounds $\alpha_{\min}, \alpha_{\max}$, trust budget β, and thresholds $\tau_{\mathrm{ECE}}, \tau_{\Delta}$
for each epoch or evaluation window **do**
  Compute calibration diagnostics ECE_t and uncertainty $\hat{\sigma}_{\text{verifier}}$ on recent held-out rollouts
  Compute shaping authority $\alpha_t \leftarrow \mathrm{clip}(\alpha_0 \cdot \sigma(\kappa(1 - \mathrm{ECE}_t)), \alpha_{\min}, \alpha_{\max})$
  Compute trajectory weights $w(\tau) \leftarrow \min(1, \beta / (\hat{\sigma}_{\text{verifier}}(\tau) + \epsilon))$
  Apply authority to updates (soft weighting; avoid hard gating unless calibrated)
  Measure drift signals: planner–policy disagreement Δ_t, memory inconsistency, synthetic-real divergence
  if $\mathrm{ECE}_t > \tau_{\mathrm{ECE}}$ **or** $\Delta_t > \tau_{\Delta}$ **or** drift diagnostics spike **then**
    Roll back authority: $\alpha_t \leftarrow \alpha_{\min}$ and cap $w(\tau)$ more aggressively
    Increase grounding weight of environment reward and reduce synthetic replay quota
  end if
end for

3.6. Operator Composition in Practice

Real systems rarely deploy a single operator in isolation. Planner outputs influence action priors, verifier outputs influence update trust, and world outputs influence replay composition at the same time. Composition creates leverage because each operator can compensate weaknesses of another: verifiers can reject planner-induced shortcuts; planners can constrain expensive search; world models can pre-screen poor branches before real interaction. Composition also creates coupling risk because correlated errors can align in harmful ways. For example, a planner may generate semantically

 doi:10.20944/preprints202603.0229.v1

attractive but infeasible trajectories, while a verifier trained on linguistic coherence assigns them high confidence, and a simulator reproduces the same bias in synthetic replay.

To manage coupling risk, practical pipelines increasingly adopt staged composition. Early training uses conservative authority, with planner guidance and verifier weighting applied softly while environment reward dominates updates. As calibration evidence accumulates, module influence is increased in states where disagreement is low and uncertainty is bounded. During deployment or shift-sensitive phases, influence can be reduced dynamically when diagnostics indicate channel drift. This staged approach is less elegant than fixed-policy integration but has better empirical stability in open-ended tasks.

A second practical pattern is counterfactual channel testing. Instead of only evaluating final policy quality, systems are tested under channel ablation and perturbation: planner disabled, verifier noise injected, synthetic replay removed, or tool outputs delayed. The resulting performance deltas reveal which pathways are essential and which are brittle. Such diagnostics are especially important for multi-module systems where nominal gains may be driven by hidden dependencies that do not transfer across environments.

Algorithm 2 Generic LLM-Enhanced RL Loop

Initialize policy parameters θ, module parameters ϕ, replay buffer $\mathcal{D}$
for each training iteration **do**
  Observe current context h_t (state history, tool traces, memory)
  Generate module outputs $o_t \leftarrow M_\phi(h_t)$
  Compose intervention channels (planner, reward/verifier, world, memory)
  Roll out policy π_θ in environment with channel-conditioned control
  Compute grounded and auxiliary supervision (e.g., r_t, b_t, $w(\tau)$)
  Update replay $\mathcal{D} \leftarrow \mathcal{D} \cup \{\tau\}$ and optional synthetic samples
  Update θ with weighted objective and update ϕ with calibration constraints
end for

3.7. Enabling Techniques and Scope Boundary

Some adjacent literature develops RL procedures primarily for improving language models themselves. Those methods are outside the main taxonomy of this survey unless their mechanisms are reused as modules that directly improve RL agents in environment-grounded tasks. In practice, transferable ideas include verifier calibration, process supervision interfaces, and trajectory-level weighting schemes that can be reinterpreted as reward-verifier components in LLM-to-RL systems. This boundary preserves directional clarity while still acknowledging useful enabling techniques.

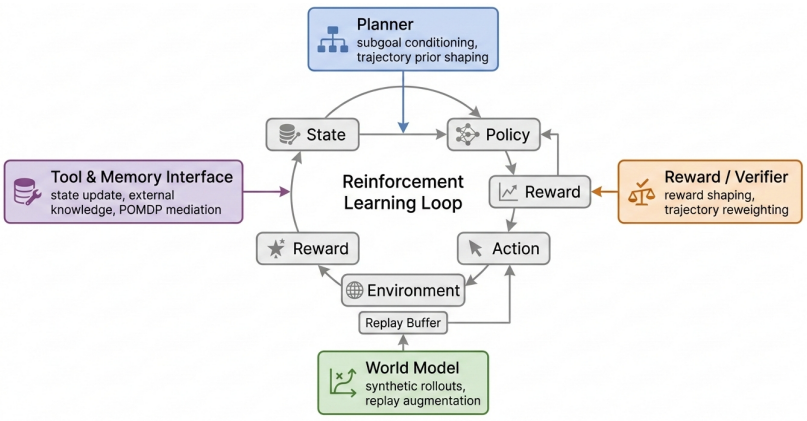

Figure 1. LLM modules embedded in the reinforcement learning loop. Planner, reward-verifier, world-model, and tool-memory modules intervene in trajectory generation and update construction while environment reward remains the grounding anchor.

 doi:10.20944/preprints202603.0229.v1

Table 1 acts as a pathway map for the rest of the survey: trajectory-prior and search modules primarily reshape sampling, reward-verifier modules reshape targets, generator modules reshape replay support, and tool-memory modules reshape effective state. This operator-centric reading improves taxonomy rigor because it links each category to a distinct optimization pathway and therefore to distinct diagnostics.

Table 1. Mechanism-first taxonomy of LLM modules in the RL loop.

Category	Intervention point	Typical signals / outputs	Typical RL setting	Rep. refs
Trajectory Prior Shaping	Action selection and option interface	Subgoals, plan sketches, option proposals	Hierarchical RL, long-horizon control	[1,6]
Supervision Restructuring	Reward and update target channels	Reward code, critique, verifier scores	Sparse-reward online RL, process-supervised RL	[4,20]
State & World Construction	Model/replay path and latent state	Imagined rollouts, synthetic transitions, abstractions	Model-based RL, expensive interaction regimes	[15,16]
Search & Deliberation Hybrids	Planning-time branching and pruning	Branch priors, candidate rationales, scoring	Tree-search control, strategic tasks	[33,34]
Data & Offline Augmentation	Dataset and replay composition	Synthetic demos, relabeling, trajectory imputation	Offline RL, low-interaction settings	[27,35]
Tool & Memory Mediation	State construction under partial observability	Retrieval context, API outputs, persistent memory	Web agents, long-context embodied interaction	[5,11]
Multi-agent / Interactive	Coordination and social credit pathways	Shared plans, negotiation feedback, role traces	Team RL and interactive planning	[36,37]

Table 2. Coverage map for taxonomy categories (part I).

ID	Category	Description	Papers (citations)
DM-A	Trajectory Prior Shaping	High-level decomposition and option proposals that shape exploration and action priors.	[1,9–11,23,24,26,28,38–64]
RV-A	Supervision Restructuring	Reward shaping, process scoring, and verifier-mediated update weighting.	[20,65]
GN-A	State & World Construction	Language-assisted state abstraction, world support, and imagined transition pathways.	[2–4,6–8,15–18,22,31,32,66–85]
DM-B	Search & Deliberation Hybrids	Tree-search or branching pipelines guided by language priors and RL grounding.	[19,21,34,86–88]

Table 3. Coverage map for taxonomy categories (part II).

ID	Category	Description	Papers (citations)
GN-B	Data & Offline Augmentation	Synthetic trajectories, relabeling, and data curation for offline or low-interaction RL.	[27,35,89–97]
IP-A	Tool & Memory Mediation	Retrieval, tool outputs, and persistent memory as augmented state channels.	[5,12–14,25,29,30,98–110]
DM-C	Multi-agent / Interactive	Coordination, communication, and interactive credit assignment under coupled adaptation.	[33,36,37,111–125]

3.8. Boundary Between Information Processor and Decision Maker Roles

The distinction between information processors and decision makers is operational, not stylistic. Information-processor modules primarily transform observations into augmented state variables s_t^{aug} while leaving both action-selection and gradient targets unchanged. Decision-maker modules, by contrast, directly influence action probabilities through a_t or latent control variables z_t and therefore change

 doi:10.20944/preprints202603.0229.v1

trajectory distribution at sampling time. Reward-verifier modules define a third boundary: they do not necessarily choose actions, but they alter update targets or trajectory weights and thus reshape gradients. This three-way split is essential for clean ablations because state mediation, action mediation, and target mediation produce different bias pathways even when they share a language backbone [1,5,19,33].

Table 4. Case-study mapping matrix linking representative systems to operator roles, authority levels, and arbitration patterns.

Representative system	Domain	Operator roles	Authority	Arbitration mechanism	Known failure mode / check
SayCan [1]	Robotics	Planner + grounded skill selection	Advisor/Controller	Prior + grounded skill-value scoring	Planner–controller mismatch; feasibility and success ablations
Code as Policies [2]	Robotics/tools	Policy/program + tool interface	Controller	Typed programs with fallback primitives	Tool execution errors; schema and behavior-level audits
Voyager [3]	Open-world	Planning + memory + curriculum	Advisor	Iterative skill updates with environment feedback	Skill drift; progress diagnostics over long horizons
Eureka [4]	Reward design	Reward synthesis + refinement	Judge (reward)	Candidate reward proposal with grounded selection	Proxy reward; reward-function ablations and transfer checks
Tool-memory agents [5]	Web/interactive	Info processor + tool/memory mediation	Advisor/Judge	Retrieval and memory policies gate context	Memory contamination; tool-failure and consistency stress tests
Search-guided hybrids [33,34]	Planning/search	Decision + verifier scoring	Controller/Judge	Branch expansion + value/verifier reranking	Prior collapse; branch-efficiency and perturbation tests
Synthetic replay pipelines [16,27]	Low-data RL	Generator + world-model assistance	Simulator (bounded)	Confidence-aware synthetic-real mixing	Model bias drift; divergence monitoring and quota sweeps

4. LLM as Information Processor

Information processing is the least noisy yet most structurally consequential role of large language models in reinforcement learning. In many practical tasks, the raw observation stream is not the state that should drive policy optimization. Useful state must be constructed from instructions, tool outputs, interaction logs, sparse sensory events, and latent constraints that are only partially explicit in the environment interface. LLM modules can function as semantic transducers that map these heterogeneous signals to compact state variables, goal descriptors, and relevance-weighted context windows. In this role, the language model does not have to choose every low-level action. Instead, it determines what information enters the control and value-estimation pathways, which can substantially change both sample efficiency and stability.

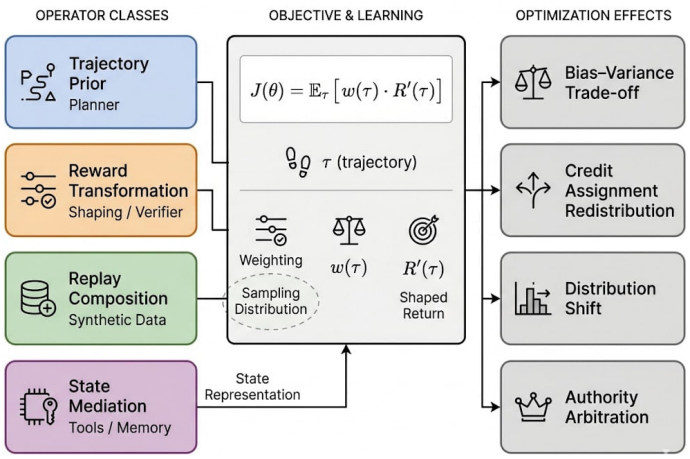

Figure 2. Role-based taxonomy of LLM-enhanced RL: information processor, reward-verifier, decision maker, and generator modules with representative submechanisms.

 doi:10.20944/preprints202603.0229.v1

This role is best understood as a representational intervention in trajectory optimization. If observation-to-state mapping improves, policy gradients are estimated with lower variance and with fewer misleading state aliases. If mapping degrades, value estimation may become systematically biased even when optimization itself is numerically stable. The practical implication is that representation design is not a preprocessing convenience; it is part of the RL algorithmic core when language modules are present. Evidence from embodied control, web interaction, and instruction-grounded decision tasks consistently supports this claim, especially in settings with long horizons and partial observability [66,67,72,99,107,109].

Unlike reward-verifier pathways that alter gradient targets directly, information-processing pathways usually improve RL by reducing state ambiguity and improving observability before updates are computed [4,19,20]. In contrast to planner-heavy systems that modify action priors online, these methods often preserve controller autonomy while still improving sample efficiency through better context structuring [1,3,33].

4.1. Language-to-State Translation and Instruction Grounding

Language-to-state modules transform natural-language goals and constraints into policy-consumable state variables. This can include latent goal vectors, symbolic predicates, option masks, or executable control hints that parameterize downstream RL components. In instruction-following environments, this translation is often the difference between random exploration and structured interaction. In embodied robotics, grounding instruction semantics into affordance-aware representations has proven especially effective because it narrows the action manifold before expensive real-world trial-and-error begins [1,2,70].

A key design choice is whether translation is static or iterative. Static translation produces one task embedding at episode start, which is computationally efficient and reproducible, but brittle under context drift. Iterative translation updates task representations as observations and tool responses arrive, enabling adaptation in dynamic interfaces at the cost of additional compute and consistency risk. Systems in web and dialogue-heavy control tasks increasingly favor iterative translation because objectives and constraints are revealed gradually, not all at once [14,71,74].

Recent embodied and interactive studies further show that iterative grounding is most useful when paired with explicit tool contracts and consistency checks, because unbounded context updates can introduce latent drift even when immediate task success appears to improve [5,11,106].

Granularity is another axis. Coarse representations improve robustness and reduce overfitting to surface wording. Rich symbolic structures can improve long-horizon decomposition when planners and verifiers are available. The practical frontier is adaptive granularity: fine-grained when uncertainty is high and coarse when control confidence is already strong.

Three design patterns recur across this line of work. First, closed-loop grounding treats translation as a controlled process: proposed task variables are validated against tool feedback or environment affordances before being committed as state [5,14,106]. Second, interface-aware grounding exploits typed tool responses and execution traces to reduce ambiguity in multi-turn settings [102,103,108]. Third, robustness-oriented grounding explicitly tests paraphrase and context perturbations rather than assuming prompt invariance [78,99,107]. Failure modes mirror these patterns: parsing drift under context accumulation, brittle reliance on tool availability, and subtle leakage when translation implicitly encodes evaluation artifacts.

4.2. Reward-Relevant Concept Bottlenecks

LLMs can also expose reward-relevant concepts that are not explicit in the native state space. In sparse-reward tasks, many transitions are weakly informative for policy updates. Concept bottlenecks introduce intermediate semantic variables such as progress indicators, safety states, or constraint-violation tags. These variables can be used in auxiliary losses, replay prioritization, or gated policy updates.

 doi:10.20944/preprints202603.0229.v1

The principal benefit is denser, semantically meaningful supervision. The principal risk is causal mismatch. A concept that predicts reward in one distribution may become spurious under shift. Robust designs therefore combine concept extraction with grounded verification against delayed environment outcomes and with periodic concept-ablation checks [31,68,77,79].

Concept bottlenecks are especially valuable when safety constraints are expressed in natural language but must be operationalized for control optimization. They improve interpretability, but only when accompanied by explicit uncertainty estimates and failure-aware arbitration.

Concept bottlenecks also bridge information processing and reward-verifier channels. When concepts are used to construct auxiliary rewards or process critics, they can accelerate learning while increasing proxy risk if the concept becomes easier to optimize than grounded utility [4,19,20]. Evaluation should therefore report both concept stability under perturbations and its correlation with delayed return, rather than only downstream performance [84,88,99].

4.3. Memory and Retrieval as State Augmentation

Long-horizon RL in interactive environments requires persistent context. Memory-mediated pipelines maintain structured traces of prior observations, tool calls, and intermediate plans, then retrieve or summarize those traces at decision time. In formal terms, the agent optimizes over an augmented state process where memory is part of the Markovian approximation. In practice, this often reduces compounding errors caused by missing context and improves recovery from earlier mistakes [5,11–13].

Retrieval-centric systems preserve detail but can become latency-heavy and noisy. Summarization-centric systems are efficient but risk irreversible information loss. Hybrid systems that combine short-term buffers with long-term summaries and verifier-guided memory correction provide better stability in multi-turn tasks.

Compared with synthetic replay methods that expand support through generated trajectories, memory-mediated systems primarily alter what is remembered and reused from grounded interaction histories [27,94,96]. This distinction matters because memory errors propagate as context bias, whereas synthetic errors propagate as transition-model bias.

An unresolved credit-assignment issue is memory write attribution: which stored item deserves credit for future return improvements. Existing methods still rely heavily on heuristics. Treating memory writes as explicit actions with delayed utility is a promising direction for principled optimization.

Design patterns here are increasingly explicit about governance. Provenance tags and typed writes reduce silent contamination, while retrieval gating limits the influence of low-confidence context [12,13,108]. Tool-aware retrieval uses execution traces and API schemas to separate environment state from external context, improving debuggability [14,101,102]. Common failure modes include stale memory persistence, tool-error amplification, and retrieval-induced shortcut learning; accordingly, evaluation should track memory drift and tool-failure recovery alongside return [5,11,106].

4.4. Programmatic and Symbolic Interface Construction

A fourth information-processing pathway uses LLMs to construct typed interfaces among language, tools, and control modules. Instead of passing free-form text, these systems emit function schemas, symbolic action arguments, or program fragments that constrain downstream execution. Such interfaces can reduce ambiguity, improve safety checks, and simplify action grounding in complex tool environments.

Programmatic mediation also improves diagnostics. Failure attribution becomes tractable when errors can be localized to parsing, grounding, tool execution, or low-level policy adaptation. This does not eliminate failures; it makes them measurable. The trade-off is reduced flexibility when schemas are too rigid. Modern systems address this with typed fallback channels and explicit exception handling [26,63,81,83].

 doi:10.20944/preprints202603.0229.v1

This interface-centric design is increasingly coupled with structured execution traces and schema-level validation in complex agentic environments, yielding better debuggability than free-form textual mediation alone [102–104,108].

From a mechanism perspective, the key design choice is where constraints are enforced: during decoding (constrained output spaces), at runtime (validators and exception handlers), or during learning (penalties and gating). Each option shifts failure surfaces and affects credit assignment: decoding-time constraints reduce action entropy, runtime validators reduce catastrophic execution, and learning-time penalties reshape gradients [63,81,83,110]. A practical evaluation insight is to separate interface correctness (schema validity, tool success rate) from downstream return, because improvements in one do not guarantee improvements in the other.

Table 5. Information-processor pathways, intervention points, and representative studies.

Subcategory	Mechanism	RL setting	Primary benefit	Representative papers
IV-A Language-to-state grounding	Instruction parsing, goal abstraction, affordance mapping	Online embodied and web RL	Reduced exploration entropy	[1,2,70]
IV-B Concept bottlenecks	Reward-relevant semantic feature extraction	Sparse-reward and constrained RL	Denser intermediate supervision	[31,68,77]
IV-C Memory-retrieval state	Persistent context updates and selective retrieval	Multi-turn partially observable tasks	Better long-horizon consistency	[5,11,12]
IV-D Symbolic interface design	Typed tool schemas and structured control tokens	Tool-centric and programmatic control	Improved controllability and auditing	[26,63,81]

Taken together, information-processing mechanisms mostly improve RL by changing what the learner perceives as state, not by changing objective semantics directly. This usually yields stable gains when observability is the bottleneck, but limited gains when supervision is the bottleneck. The next section therefore shifts focus from state construction to supervision construction, where reward and verifier channels can accelerate learning more aggressively but with higher misalignment risk.

5. LLM as Reward Designer and Verifier

Reward and verifier pathways modify the supervision that drives RL updates. Instead of relying solely on delayed environment reward, these approaches introduce language-derived shaping terms, process scores, and trajectory-level acceptance signals. The upside is clear: denser supervision can accelerate learning and stabilize long-horizon credit assignment. The downside is equally clear: auxiliary signals can become exploitable proxies that drift from task utility. Consequently, this role requires stronger calibration and diagnostics than almost any other module class [4,19,20,51,59,65].

Compared with information-processing modules, reward-verifier modules act closer to the gradient target and therefore can produce larger gains and larger failures in fewer updates [5,12,66]. Unlike planner-heavy methods that alter sampling trajectories first and targets second, supervision-first methods alter targets immediately and rely on calibration to preserve alignment with long-horizon return [1,3,33].

5.1. Natural-Language Reward Shaping for RL Tasks

Natural-language reward shaping transforms textual task requirements into explicit reward signals or code-level reward functions. In RL environments where task constraints are easier to express linguistically than mathematically, this mechanism can reduce reward-engineering overhead and speed adaptation across task variants [4,30,31,84].

A common formulation is

$$r'(s,a,s') = r(s,a,s') + F(s,s'), \tag{14}$$

where F is a language-derived shaping term. Potential-based shaping variants preserve policy invariance under standard assumptions, but practical pipelines often add task-specific heuristics that violate

 doi:10.20944/preprints202603.0229.v1

these assumptions. Such heuristics can still be useful, yet they require explicit grounding audits and coefficient control.

Several design patterns are now recurring. First, reward code generation converts task descriptions into explicit programmatic rewards, enabling faster iteration but also creating strong proxy surfaces [4,30,31]. Second, critique- or score-based shaping treats the language module as an evaluator that supplies dense intermediate feedback, which can be integrated softly through coefficients such as α_t rather than as hard constraints [19,20,65]. Third, calibration-aware authority schedules explicitly couple shaping strength to verifier reliability, shrinking shaping influence when calibration drifts (cf. Eq. 12) [19,23,24].

Empirically, reward-generation methods are most effective in robotics and embodied tasks with sparse success signals. However, they are also where reward hacking is most dangerous: policies can maximize shaped objectives while degrading true task robustness. Effective systems therefore report shaping-ablation curves, return-correlation diagnostics, and adversarial probes.

Recent work also indicates that reward synthesis quality is tightly coupled with task decomposition quality: when decomposition is poor, even semantically plausible reward code can bias optimization toward superficial process markers [76,77,79].

Evaluation in this category should therefore be multi-layered. In addition to final return, papers should report shaped-vs-grounded correlation and sensitivity to coefficient sweeps (small changes in α should not flip outcomes if shaping is well aligned) [4,84]. Shortcut resistance tests are also essential: if a policy can satisfy the shaped objective via superficial artifacts, then shaped gains are unlikely to transfer [20,88].

5.2. Verifier-Based RL and Process Critics

Verifier-based RL does not always replace reward; it reweights or filters updates using process-level quality scores. A verifier may score full trajectories, sub-trajectory segments, or step-wise rationale traces. These scores enter optimization via weighting or gating, typically to suppress noisy samples and improve update quality [19–21,24].

The main technical challenge is calibration under shift. A verifier that appears reliable in one prompt regime can fail when task framing changes. Over-authorized verifiers induce shortcut optimization, where the policy learns to satisfy verifier artifacts instead of environment objectives. Practical mitigation includes disagreement-aware ensembles, uncertainty heads, and staged authority schedules.

A second challenge is consistency across trajectories of different length and structure. Length-normalized scoring and context-aware calibration are important in multi-turn tasks where superficial fluency is weakly tied to control quality.

Cross-domain evidence from robotics, web interaction, and strategic environments suggests that verifier transfer is weakest when score semantics depend strongly on hidden tool state; this is one reason disagreement-aware verification is becoming standard [5,14,88].

A useful way to compare verifier designs is by how the score is consumed. Soft weighting tends to degrade more gracefully than hard gating under mild calibration drift, while hard gating is only defensible when verifiers are demonstrably calibrated and monitored online [19,20]. Ensemble disagreement provides a practical uncertainty proxy and supports rollback triggers when the verifier enters out-of-distribution regimes [23,24]. For evaluation, papers should report calibration metrics (ECE/AUROC) on held-out trajectories and calibration stability under prompt/tool perturbations, not only in-distribution accuracy [19,78,99].

5.3. Process Reward Decomposition and Credit Routing

Process supervision can be interpreted as trajectory-level credit routing. Instead of rewarding only terminal outcomes, the system assigns credit to intermediate decisions based on language-mediated evaluations of progress quality. This can dramatically reduce delayed-credit ambiguity in compositional tasks.

 doi:10.20944/preprints202603.0229.v1

Yet process-level supervision can fail through local myopia: optimizing intermediate criteria may reduce final return when global dependencies are ignored. To mitigate this, modern pipelines combine process rewards with delayed grounding signals and include anti-myopia checks such as delayed reward consistency penalties [65,85,87].

Process decomposition is especially promising for hierarchical RL, where each level can receive tailored supervision. This creates a tractable route to long-horizon training but requires careful coordination to avoid conflicting gradients across hierarchy levels.

Design patterns in process supervision include segment-level scoring (local progress critics), hierarchy-aware credit routing (distinct verifiers per layer), and counterfactual consistency checks that compare process signals to delayed grounded outcomes [65,85,87]. Two common failure modes are local myopia (process targets encourage short-horizon satisfaction) and process gaming (policies learn to satisfy evaluator artifacts). Evaluation should therefore include segment-to-return correlation analyses and stress tests where process cues are perturbed while environment reward is held fixed [20,21,24].

Many recent systems therefore pair process rewards with verifier-based reweighting and calibration-aware authority schedules that damp intermediate signals when they become unreliable, reducing the risk that local process objectives override delayed grounding [19,23].

5.4. Safety-Oriented Verifier Channels

In safety-sensitive domains, verifiers often enforce constraints rather than maximize unconstrained reward. Language modules can encode procedural rules, detect violations, and trigger update suppression or fallback behavior. Hybrid designs that combine language checks with symbolic validators are generally more reliable than pure language verification [51,59,82].

Because constraint checking is most brittle when inputs are free-form, several systems wrap language checks with typed tool schemas and symbolic validators that make violations machine-testable [63,81].

The practical tension is exploration versus safety strictness. Overly strict verifiers can collapse exploration and stall learning; weak verifiers fail to prevent unsafe drift. Adaptive safety shaping based on uncertainty and state risk appears to offer a useful compromise.

In safety-oriented channels, evaluation should report both success and violation profiles. Improvements in violation rates are only meaningful if task performance does not collapse or shift toward degenerate safe behaviors. Practical protocols include staged enforcement (soft penalties early, stricter gating late) and explicit rollback policies when safety verifiers become uncertain [19,51,59,82].

To avoid either exploration collapse or silent unsafe drift, staged enforcement is increasingly combined with disagreement-triggered rollback: when ensemble disagreement or uncertainty spikes, the system reduces verifier authority and falls back to conservative controllers [23,24].

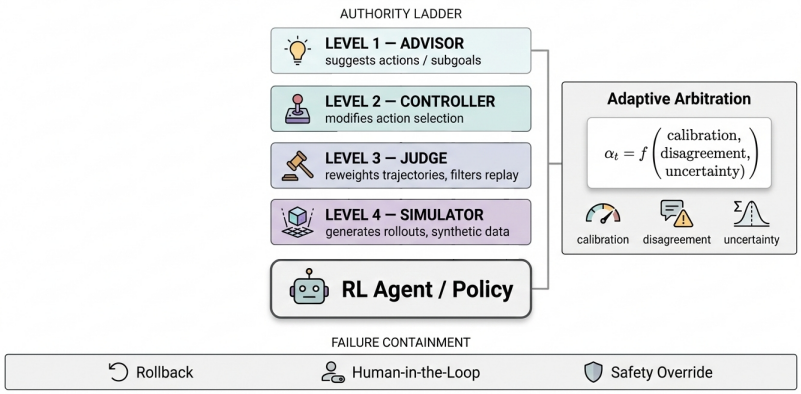

Figure 3. Supervision pathways in LLM-enhanced RL. Environment reward, shaping terms, verifier scores, and synthetic influences are arbitrated before policy updates.

 doi:10.20944/preprints202603.0229.v1

Table 6. Reward and verifier pathways with optimization effects and failure modes.

Mechanism	Where it enters	Typical gain	Typical failure	Representative papers
V-A Reward code generation	Constructs shaped reward signal	Faster adaptation in sparse reward	Proxy exploitation / misspecification	[4,30,31]
V-B Verifier weighting	Reweights updates or replay samples	Noise suppression and process control	Verifier shortcut optimization	[19,20,24]
V-C Process decomposition	Segment-level trajectory supervision	Improved long-horizon credit routing	Local optimum process gaming	[65,85,87]
V-D Safety verification	Constraint checks and gating	Lower violation rates	Over-conservative policy behavior	[51,59,82]

Reward-verifier pathways are often the highest-leverage and highest-risk intervention class. They can convert delayed outcomes into actionable intermediate targets, but they also create direct routes for proxy optimization. This tension motivates the transition to decision-making modules, where language channels act less as judges of quality and more as shapers of trajectory search.

6. LLM as Decision Maker (Planner and Policy Improver)

Decision-making interventions assign language modules direct influence over action selection, branch expansion, or exploration strategy. Compared with representation and reward mediation, these interventions generally carry higher authority. Their upside is substantial in long-horizon tasks where unstructured search is intractable. Their failure modes are also severe: planner-policy mismatch, infeasible action proposals, and overconfident deliberation under distribution shift [1,3,6,8,33,52,87].

Unlike reward-verifier pathways that primarily reshape update targets, decision-making pathways reshape sampling trajectories first and only indirectly alter gradients through changed visitation distributions. In contrast to generator pathways that modify replay support post hoc, planner and search interventions affect control online and therefore require tighter real-time fallback logic [4,19,27,35].

6.1. LLM as Planner for Hierarchical RL

Planner-centric systems decompose long-horizon tasks into subgoals or options that condition lower-level RL controllers. The immediate benefit is horizon reduction: instead of optimizing over full trajectories with delayed sparse reward, the learner receives structured intermediate objectives that can be evaluated sooner. In hierarchical form, the policy becomes conditioned on planner outputs z_t, and policy learning focuses on executing and refining those latent directives [1,3,6,8].

A critical design choice is planner authority. Soft planners provide candidate subgoals that the controller can ignore; hard planners define mandatory execution scaffolds. Soft planners are safer under shift but may underuse useful structure. Hard planners can yield faster convergence in stable settings but fail sharply when generated plans are dynamically infeasible. Hybrid authority schedules, where planner influence grows with feasibility confidence, are increasingly common.

Three additional design patterns are worth distinguishing. First, feasibility-aware planning couples proposal generation with grounded critics that reject or repair dynamically invalid subgoals, turning planning into a closed-loop contract rather than a one-way suggestion [1,6,8,9]. Second, option-centric planning emits discrete skills or executable program sketches, which improves debuggability but can reduce adaptability when schemas are too rigid [2,10,48]. Third, arbitration-centric planning explicitly ties planner influence to disagreement and calibration signals, enabling rollback when mismatch rates spike [19,20,23].

Planner failure modes are correspondingly predictable: plan feasibility mismatch, overconfident long-horizon decomposition, and brittle dependence on interface conventions. Evaluation should therefore report feasibility/repair rates and plan-to-return correlation, not only final success [6,9,33].

Planner modules also differ by output form. Some output natural-language subgoals, others output symbolic options or executable pseudo-code. Symbolic and code-like outputs can improve

 doi:10.20944/preprints202603.0229.v1

reproducibility and repairability, but they require strong schema validation. Free-form outputs are flexible but harder to audit.

Planner-centric deployment settings are also broadening. Beyond manipulation and navigation, planners are increasingly evaluated in structured driving and autonomy scenarios where latency matters and plan refinement must be overlapped with low-level control [75].

Across reported systems, symbolic planners tend to improve debuggability and contract checking, whereas free-form planners often retain higher adaptability under open-ended tasks; hybrid planners that emit structured plans with natural-language justifications are increasingly common [2,10,48,49].

6.2. LLM-Guided Search and Deliberative Control

Search-augmented decision pipelines use LLM priors to guide branch generation while RL/value estimates ground branch selection. This includes MCTS-style proposals, beam-style expansion with verifier reranking, and uncertainty-aware branch sampling [19,33,34,86].

The core benefit is better allocation of planning-time computation. Rather than evaluating actions uniformly, the agent explores branches likely to contain high-value trajectories and prunes implausible branches early. This can improve both solution quality and sample efficiency in strategic and language-rich environments.

The main risk is prior collapse: language priors can overconcentrate search on semantically attractive but environment-weak branches. Mitigation includes diversity regularization, value-grounded pruning, and adversarial branch perturbations. Another risk is latency inflation. Deliberation improves quality only if compute budgets are adapted to uncertainty; fixed over-deliberation reduces practical utility.

Recent deliberative-control results also highlight a recurring calibration pattern: branch quality estimates remain reliable at shallow depth but deteriorate quickly as branches lengthen without grounded feedback, reinforcing the need for depth-aware trust schedules [19,21,86].

Search pipelines also expose a distinct evaluation axis: how much computation is spent per unit performance gain. Reporting should include branch-efficiency metrics (expansions per solved task, pruning ratios) and latency profiles under fixed budgets [33,34,88]. Failure modes include prior collapse (semantic plausibility overwhelms grounded value), and verifier-induced narrowing when rerankers over-penalize diverse branches [19,24,87].

6.3. Policy Priors and Action Proposal Modules

A complementary family keeps a standard RL optimizer but uses LLM modules to propose candidate actions, skills, or policy priors. In continuous control and robotics, proposals may be high-level control intents or waypoint programs. In discrete environments, proposals can restrict action sets before value scoring. This narrows exploration and can reduce sample complexity.

Policy-prior methods are most effective when action semantics are structured and compositional. They are less effective when fine-grained dynamics dominate and linguistic abstractions lose predictive value. Robust implementations therefore include controller-side feasibility critics and fallback actions.

Design patterns include action-set restriction (prune the discrete candidate set before value scoring), continuous intent proposal (suggest waypoints or sub-policies for a low-level controller), and imitation-to-RL handoff where language demonstrations warm-start but are corrected by grounded feedback [46,48,49,57]. Failure modes are often subtle: priors can entrench early shortcut behaviors and reduce exploration diversity. Evaluation should therefore include early-training sensitivity analyses (how quickly priors dominate updates) and ablations that remove the proposal channel after convergence [50,51,53].

Recent variants treat proposal authority as adaptive, increasing reliance on the prior only when feasibility critics agree and reverting to baseline policies under elevated disagreement [23].

Imitation-enhanced variants use language-generated demonstrations or policy sketches to warm-start training, then rely on RL for correction. These approaches often improve early learning but can entrench prior bias if corrective feedback is weak [46,48,49,57].

 doi:10.20944/preprints202603.0229.v1

6.4. Exploration Strategy and Curriculum Control

Decision-making roles also include exploration shaping and curriculum generation policies. LLM modules can propose exploration targets, curriculum stages, and adaptive data-collection plans based on observed failure patterns. This is particularly useful in sparse-reward domains where random exploration is prohibitively expensive.

Curriculum-guided exploration is not uniformly beneficial. Poorly calibrated curricula can lock the learner into easy but uninformative trajectories. Effective systems update curriculum policies using grounded performance diagnostics and maintain diversity constraints over generated tasks [52,54,58].

Compared with reward-shaping interventions that alter gradients directly, curriculum interventions alter the data seen by the optimizer and therefore tend to improve stability when reward semantics are uncertain [4,65,87]. However, they are more sensitive to task-generation bias and require diversity-aware sampling policies [80,90,93].

Two failure modes are common: curriculum collapse (generated tasks become redundant) and competence misestimation (difficulty escalates faster than the controller can learn). Evaluation should therefore report competence curves and diversity statistics over generated tasks, not only final return [52,54,58,60].

6.5. Multi-Agent and Interactive Decision Settings

In multi-agent settings, language modules can coordinate role assignment, negotiation, and shared planning. These settings expose additional credit-assignment complexity because outcomes depend on coupled adaptation among agents. Language-mediated coordination can reduce non-stationarity when communication channels are reliable, but can also induce proxy collusion or communication drift under weak constraints [36,37,112,123].

Robotic multi-agent task planning and cooperative transport highlight an additional constraint: coordination policies must preserve role consistency and shared plan semantics under partial observability [111,122].

Evaluation here must include team-level stability, communication efficiency, and resilience to partial communication failure. Single-agent metrics are insufficient for assessing cooperative reliability.

Multi-agent settings also make verifier and memory issues more acute: communication traces become part of state and can be gamed or drift under coupled adaptation. Practical evaluations therefore include partial communication ablations and adversarial partner tests, in addition to aggregate team return [112,116,123,124].

Table 7. Decision-maker mechanisms by action space, horizon, and domain.

Decision pathway	Action space	Horizon profile	Environment family	Representative papers
VI-A Hierarchical planning	Mixed discrete or continuous	Medium to long horizon	Robotics, embodied tasks	[1,3,6]
VI-B Search guidance	Primarily discrete branches	Medium to long horizon	Strategy and reasoning environments	[33,34,87]
VI-C Policy prior proposals	Continuous or hybrid	Short to medium horizon	Continuous control and recommendation	[46,48,57]
VI-D Exploration and curriculum control	Any	Long training horizon	Sparse-reward open-ended RL	[52,54,58]
VI-E Multi-agent coordination	Mixed	Medium to long horizon	Team coordination and interactive tasks	[36,37,123]

Decision-making modules expose a core trade-off between guidance and controllability: stronger priors reduce search burden but increase dependence on planner quality. This naturally leads to generator pathways, where language modules influence not only action selection but also the data and world structures from which policies learn.

 doi:10.20944/preprints202603.0229.v1

7. LLM as Generator (World Models, Data, and Environment Design)

Generator roles alter the data distribution that RL learns from. Instead of only guiding actions or rewards, these modules generate imagined transitions, synthetic trajectories, relabeled datasets, and auto-generated tasks. This can dramatically increase coverage in low-data regimes and accelerate early-stage learning. It can also inject structured bias that remains hidden until deployment shift [15–17,27,35,90,96].

In contrast to decision-making modules that alter trajectories online, generator modules often alter learning indirectly by changing replay composition and world assumptions. Compared with information processors, they can yield larger sample-efficiency gains in low-interaction settings but face stronger distribution-shift risks that demand explicit uncertainty control [5,11,28,33].

7.1. World-Model and Simulator Generation

Language-assisted world modeling uses LLM priors to produce abstract or explicit transition hypotheses. In model-based RL, these hypotheses support planning and value-target construction. In expensive interaction settings, even partially accurate world priors can improve sample efficiency by reducing random exploration [15–17,32].

In practice, world-model generators are most reliable when paired with grounded correction channels rather than treated as stand-alone simulators. One common pattern is abstraction-first modeling, where the language module proposes coarse latent state summaries or event structure while a learned dynamics model or environment interface supplies low-level transitions. A second pattern treats the language generator as a proposal distribution: it emits candidate next states or rollout sketches that are accepted only after feasibility checks or value-based consistency screening. A third pattern is disagreement-gated imagination, where multiple prompts or auxiliary models produce alternative rollouts and only high-consensus transitions are admitted into replay [15,16,18,29].

The dominant risk is hallucinated dynamics. Language-generated trajectories can be internally coherent but physically invalid. Without uncertainty-aware arbitration, these errors contaminate replay and value learning. Strong systems therefore cap synthetic replay ratios, monitor synthetic-real divergence, and discount deep imagined rollouts more aggressively than shallow ones.

Beyond hallucination, a second failure mode is self-confirming model bias: policies can learn to exploit systematic world-model errors, producing inflated imagined returns that collapse under grounded evaluation. This interaction is most likely when synthetic rollouts dominate updates or when the planner's branching policy overfits to language priors. Mitigations include value-aware rollback triggers, periodic re-anchoring to environment reward, capped rollout depths, and conservative schedules that raise synthetic influence only after stability is observed [16,17,32,35].

Reported best practices now include per-transition confidence tagging and replay-time filtering rather than one-shot dataset filtering, because transition quality is often non-uniform within a single generated trajectory [18,32,81].

Evaluation should therefore treat world-model quality as an intermediate variable, not an implicit assumption. In addition to outcome metrics, reports can include multi-step prediction error on held-out interactions, divergence between synthetic and real state-feature distributions, and ablations over rollout depth and synthetic mixing ratio. A useful sanity check is to compare improvements when synthetic rollouts are shuffled or replaced by grounded samples; large sensitivity suggests the system is learning from generator artifacts rather than from transferable dynamics structure [16,17,29].

7.2. Synthetic Trajectory and Demonstration Generation

Synthetic data generation includes demonstrations, candidate trajectories, and relabeled traces. These resources are valuable when real data is costly or unsafe to collect. Offline RL pipelines in particular benefit from synthetic augmentation when data support is narrow.

Yet generation quality is not binary; it is distributional. A dataset can contain many plausible trajectories that still bias learning toward benchmark-specific shortcuts. Consequently, filtering and

 doi:10.20944/preprints202603.0229.v1

provenance are as important as generation. High-quality pipelines attach confidence metadata, enforce split hygiene, and report performance sensitivity to synthetic-data removal [27,28,35,89].

Other generator-centric pipelines emphasize that synthetic datasets should be governed as first-class artifacts with explicit provenance and contamination tests, rather than treated as implicitly trustworthy demonstrations [91].

A first design pattern uses generated demonstrations primarily for cold start, after which grounded RL gradients take over; mixing schedules taper synthetic influence as on-policy data grows. A second pattern synthesizes counterfactual trajectories conditioned on alternative goals or failure explanations, effectively performing semantic relabeling that can improve generalization when sparse rewards underspecify progress. A third pattern couples generation with verifier-based filtering: candidate traces are scored for feasibility, safety, or goal satisfaction and then either selectively admitted into replay or softly reweighted [27,35,89,94,96].

Two failure modes recur. First, synthetic confounding: generated traces encode stylistic cues correlated with benchmark scoring rather than causal control structure, causing optimistic offline evaluation. Second, distributional mode collapse: generators overproduce a narrow set of "successful" traces, reducing coverage and harming exploration once deployed online. Mitigations include diversity constraints, split-hygiene audits, and influence tracking that estimates how much policy improvement depends on generated versus grounded support [89,94,97].

Evaluation should therefore report sensitivity curves over synthetic mixing ratio and filtering threshold, and should separate "data benefit" (support expansion) from "label benefit" (improved supervision) by holding one constant while varying the other. When available, online fine-tuning after offline pretraining serves as a grounded reality check: if online returns regress despite strong offline scores, synthetic confounding or over-optimistic support assumptions are likely [27,28,35].

7.3. Auto-Curriculum and Environment Generation

Auto-curriculum generators create task sequences and difficulty schedules tailored to policy progress. Environment generators create new scenarios for robustness training and stress testing. Together, they can improve generalization by broadening the training manifold and preventing overfitting to static benchmark distributions.

The challenge is maintaining useful diversity. If generated tasks are too similar, curriculum reduces to redundancy. If they are too difficult too early, optimization destabilizes. Adaptive curriculum policies should therefore condition on measured competence and uncertainty, not only on generator confidence [80,90,93].

A useful conceptual split is between competence-driven curricula (generate tasks at the frontier of current capability) and coverage-driven curricula (generate tasks that expand state visitation). Competence-driven schemes often rely on success-rate histories, while coverage-driven schemes use novelty heuristics or disagreement proxies. A third pattern is adversarial task generation: generators seek failure cases to harden policies, but must be constrained to avoid degenerate "gotcha" tasks that do not transfer [52,54,58,90].

Several recent curricula further implement staged escalation with explicit competence models and rollback triggers when performance degrades or uncertainty spikes, which helps prevent destabilizing difficulty overshoot [60].

Curriculum failure modes are correspondingly structural: redundancy collapse, difficulty overshoot (destabilization and forgetting), and generator bias that overfits to benchmark artifacts. Evaluation should therefore report competence curves, diversity/coverage statistics, and cross-task generalization under a held-out task set. Environment generation also connects to procedural content generation and transfer considerations: curricula that optimize short-term return can still harm transfer if they overfit to generator artifacts [64].

 doi:10.20944/preprints202603.0229.v1

7.4. Offline Relabeling and Curation

In offline RL, LLMs can annotate trajectories with latent goals, quality labels, and process summaries. These annotations can improve conservative policy learning by exposing otherwise hidden structure in historical data. They can also introduce confounding if labels reflect language priors not grounded in dynamics.

Reliable offline augmentation requires conservative integration schedules and explicit uncertainty modeling. Policies should be evaluated under multiple relabeling strengths and with diagnostics that isolate label quality effects from policy-optimization effects.

Three practical patterns have emerged. First, multi-granularity annotation attaches both trajectory-level summaries (goal, outcome, rationale) and transition-level tags (subgoal completion, constraint violations), enabling downstream learners to choose the supervision timescale that best matches the control horizon. Second, conservative integration treats labels as soft auxiliary targets or reweighting signals rather than as hard replacements for reward, reducing the risk of mislabel-induced objective drift. Third, uncertainty gating and disagreement filtering restrict label usage to high-confidence segments, which is particularly important when historical data spans heterogeneous policies or logging regimes [35,91,94,96].

Two failure modes are especially common: label-induced confounding (annotations correlate with surface cues rather than with controllable factors) and support mismatch (labels shift the effective dataset distribution toward regions where value estimates are unreliable). Mitigation requires ablations over relabeling strength, counterfactual contamination tests, and evaluation under explicit distribution-shift suites rather than only under i.i.d. test splits [89,92,95].

Unlike reward-verifier pathways that can be stress-tested with calibration curves on held-out trajectories, offline augmentation needs contamination diagnostics that estimate how much policy gain depends on generated versus grounded support [89,94,97].

This distinction explains why many offline gains are fragile under distribution shift: improvements can come from improved support coverage or from latent confounding, and only counterfactual contamination analyses can separate the two effects [35,91,92,95].

Table 8. Generator pathways and their benefit-risk profile across RL settings.

Generator role	Generated artifact	Where it helps	Where it can hurt	Representative papers
VII-A World-model assistance	Imagined transitions and rollout abstractions	Expensive interaction and long-horizon planning	Hallucinated dynamics and value drift	[15–17]
VII-B Synthetic demonstrations	Candidate expert traces and trajectory samples	Cold start and sparse real data	Synthetic confounding and leakage	[27,35,89]
VII-C Auto-curriculum generation	Task sequences and difficulty schedules	Open-ended and sparse-reward training	Curriculum collapse / poor diversity	[52,80,93]
VII-D Offline relabeling and filtering	Goal labels, quality tags, process summaries	Support expansion in offline RL	Mislabel bias and support mismatch	[94,96,97]

Across the four module families, a consistent systems pattern emerges. Information processors mainly improve observability and representation quality; reward-verifier channels mainly reshape supervision density; decision-maker channels mainly reshape trajectory priors; and generator channels mainly reshape data support. High-performing systems rarely rely on only one family. Instead, they combine moderate planner authority, calibrated verifier guidance, conservative synthetic augmentation, and tool-memory mediation with explicit arbitration logic. This composition strategy explains why apparently different architectures often converge to similar intervention contracts.

The same cross-family view clarifies transfer behavior. Methods that gain primarily through representation and memory often transfer better across interface shifts, because they alter state construction rather than objective semantics. Methods that gain primarily through reward shaping or strong verifier gating often show larger in-domain improvements but higher sensitivity to proxy mismatch. Methods that gain through synthetic generation can scale quickly in low-data settings but require the strongest uncertainty controls. Evaluating systems by this intervention profile, rather than by module branding, yields more reproducible design guidance.

 doi:10.20944/preprints202603.0229.v1

8. Evaluation, Benchmarks, and Practical Guidelines

Evaluation is a central bottleneck for this field because modules intervene at different points of the RL loop while papers still report mostly outcome-level metrics. Without pathway-aware diagnostics, strong reported gains may be driven by fragile contracts that fail under shift. A coherent evaluation ecosystem should therefore jointly measure final performance and supervision integrity.

8.1. Benchmark Families

Current evaluations span robotics and embodied control, web and interactive agents, games and language-rich environments, continuous control benchmarks, and multi-agent coordination tasks. Each family stresses different pathway failures. Robotics emphasizes dynamics fidelity and safety constraints. Web tasks emphasize tool reliability, memory consistency, and recovery from interface noise. Strategic game-like settings emphasize long-horizon planning and branch allocation. Offline datasets emphasize support mismatch and conservative generalization.

Embodied evaluations are also expanding from fixed tasks toward development-style benchmarks in which agents must iteratively build or improve embodied behaviors, exposing long-horizon tool and memory failure modes that simple success-rate reporting can mask [55].

Benchmark fragmentation is compounded by heterogeneous implementation contracts. Prompt formats, retrieval budgets, memory truncation policies, and verifier thresholds can materially alter outcomes while remaining underreported. Standardized benchmark cards should include these factors as first-class metadata, not implementation footnotes.

8.2. Metrics and Pathway Diagnostics

Outcome metrics remain necessary: return, success rate, sample efficiency, and constraint violation rates. However, they should be complemented with channel diagnostics: planner feasibility, verifier calibration, synthetic-real divergence, memory freshness, tool failure rates, and arbitration-switch frequency. These diagnostics reveal causal pathways for gains and failures.

Robustness metrics are essential for deployment-oriented evaluation. Useful stress tests include prompt paraphrase perturbations, tool outages, adversarial verifier probes, and delayed-reward perturbations. Reporting only in-distribution metrics risks overestimating practical reliability.

The checklist in Table 9 is most informative when diagnostics are reported jointly rather than independently. For example, a rise in planner-policy disagreement is less concerning when verifier calibration and memory drift remain stable, but becomes high risk when both degrade simultaneously. Similarly, synthetic-real divergence should be interpreted together with replay contamination ratio: high divergence with low synthetic influence may be tolerable, while moderate divergence with dominant synthetic influence is usually a warning condition [5,16,27,96]. Cross-diagnostic reporting therefore improves causal interpretation of gains and helps avoid overfitting to single-metric optimizations.

To make the checklist operational rather than descriptive, studies should also report which corrective action is taken when a diagnostic crosses a warning threshold. As a concrete example, rising ECE or collapsing AUROC should trigger verifier trust contraction (reduce weighting, increase disagreement gating, or fall back to environment reward for updates); persistent synthetic-real divergence should tighten synthetic quotas and shorten rollout depths; sustained planner-controller mismatch should reduce planner authority and increase repair frequency; and tool-memory drift should trigger context quarantine and re-grounding rather than continued accumulation. Reporting these conditional interventions alongside outcome metrics helps distinguish robust algorithmic improvements from fragile orchestration choices [1,5,8,16,19,20].

A practical rollout strategy is to gate deployment by checklist stability rather than by return alone: require verifier calibration to remain within tolerance, planner-policy disagreement to trend downward, and synthetic contamination to remain bounded over multi-seed runs. This protocol improves practical usefulness by turning abstract diagnostics into release criteria.

 doi:10.20944/preprints202603.0229.v1

Table 9. Evaluation checklist for pathway-level diagnostics in LLM-enhanced RL systems.

Diagnostic	Metric	How to compute	Threshold / warning sign	Relevant operator
Verifier calibration	ECE, AUROC	Bin verifier confidence on held-out trajectories; compute confidence-accuracy gap and discrimination under success/failure labels	ECE $>$ 0.08 or rapidly degrading AUROC indicates unsafe verifier authority	Reward & Verifier [19,20,24]
Synthetic-real divergence	MMD/KL proxy + TD-residual gap	Compare feature distributions between synthetic and grounded replay; track value-error gap across the two buffers	Persistent divergence growth or TD gap inflation flags model-bias accumulation	World and Generator [16,17,27]
Planner-policy disagreement rate	Feasibility mismatch ratio	Measure frequency that planner proposals are rejected or repaired by controller/value critic	Sustained mismatch $>$ 20% suggests planner authority should be reduced	Planner and Decision modules [1,6,8]
Replay contamination ratio	Synthetic influence fraction	Estimate contribution of synthetic transitions to policy gradient norm or update mass	High influence with weak grounded gains indicates shortcut learning risk	Offline and Generator channels [35,89,96]
Memory drift	Context inconsistency rate	Track contradiction/staleness in retrieved memory versus current environment state and tool outputs	Rising inconsistency under stable task conditions signals context-governance failure	Tool and Memory channels [5,12,13]
Tool-call reliability	Timeout/error rate, latency	Log tool-call failures and latency distributions; measure recovery success under outages	Error spikes or heavy tails indicate brittle tool dependence	Tool and Memory channels [5,14,106]
Authority escalation stress test	Return vs authority curves	Sweep α and weight caps; report stability and rollback frequency under shift perturbations	Sharp cliffs suggest over-authorized channels	Arbitration/All operators [19,20,33]
Distribution-shift robustness	Stress-suite pass rate	Evaluate under prompt paraphrase, tool outages, altered dynamics, and delayed reward perturbations	Large drops without diagnostic warnings indicate weak monitoring	All operators [78,88,99]

8.3. Reproducibility and Ablation Design

Ablations should align with claimed mechanisms. Planner-centric systems should ablate planner authority and repair policies. Reward-centric systems should ablate shaping strength and verifier weighting. Generator-centric systems should ablate synthetic-real mixing ratios and confidence filters. Memory-centric systems should ablate retention windows and retrieval quality.

A second reproducibility requirement is intervention logging. Systems should log module outputs, confidence values, arbitration decisions, and tool interactions. Without these traces, many failures are not reproducible and mitigation claims are difficult to verify.

8.4. Practical Design Patterns

Across domains, several design patterns recur. First, stage authority over training phases: conservative module influence early, increased influence after calibration evidence. Second, maintain dual grounding checks that compare language-mediated signals against environment return. Third, keep fail-safe fallback policies that disable high-risk channels under disagreement spikes.

These patterns are not universal guarantees, but they consistently improve robustness in reported deployments. They also provide a path toward standardized engineering practice for LLM-enhanced RL [1,5,16,19,51].

To make these patterns comparable across papers, reporting should separate intervention efficacy from orchestration efficacy. Intervention efficacy measures whether a channel (e.g., planner or verifier) can provide useful signal in principle. Orchestration efficacy measures whether authority schedules, fallback logic, and arbitration thresholds are tuned robustly in practice. Confusing the two leads to misleading conclusions: a weakly orchestrated strong intervention may appear inferior to a well-orchestrated weaker one. Standard reports should therefore include both mechanism ablations and orchestration ablations, with perturbation tests on tool reliability, prompt framing, and uncertainty calibration.

In practice, this separation also improves reproducibility audits: when a result regresses, one can determine whether the failure came from weakened module signal quality or from altered orchestration policy, rather than conflating both as "model instability."

 doi:10.20944/preprints202603.0229.v1

Table 10. Benchmark families, metrics, and high-frequency evaluation pitfalls.

Domain family	Typical tasks	Core metrics	Frequent pitfall	Representative papers
Embodied robotics	Manipulation, navigation, assembly	Success, return, safety violations	Sim-to-real mismatch and brittle shaping	[1,16,32]
Web and interactive agents	Multi-step browsing and tool workflows	Task completion, step cost, latency	Tool variance and memory drift	[5,14,106]
Games and strategy	Long-horizon sparse-reward control	Return, win rate, branch efficiency	Prompt leakage and prior collapse	[3,33,88]
Offline RL	Policy learning from static data	Normalized return, OOD robustness	Synthetic confounding and support mismatch	[27,35,89]
Multi-agent systems	Coordination and role allocation	Team return, coordination stability	Non-stationarity and proxy collusion	[36,37,123]

Taken together, the benchmark families and diagnostic checklist suggest a shift in evaluation culture: LLM-enhanced RL systems should be judged not only by return or success rate, but by whether their supervision channels remain stable under controlled perturbations. Reporting authority sweeps, synthetic mixing sensitivity, tool outage tests, and calibration drift under prompt variation makes it possible to anticipate which systems will fail gracefully and which will fail catastrophically under deployment shift [16,19,99,107].

9. Failure Modes and System-Level Trade-Offs

The most persistent failures in LLM-enhanced RL are not isolated module mistakes; they are interaction failures across supervision channels. Planner proposals can be semantically valid yet dynamically infeasible. Verifier channels can be locally calibrated yet globally misaligned with long-horizon return. Synthetic data channels can improve early coverage while silently shifting value targets away from environment-grounded behavior. Tool-memory pathways can improve observability while propagating stale context over many decision steps. Each failure originates in a distinct contract violation: authority granted without calibrated uncertainty, integration performed without counterfactual checks, or channel output consumed outside its reliability regime.

A practical implication is that mitigation should be contract-aware rather than module-specific. The same planner output may be safe as advisory context but unsafe as hard control. The same verifier score may help as soft weighting but harm as binary gating. The same synthetic replay may improve early value shaping but destabilize late training if replay mixing is not annealed. Effective systems therefore combine confidence-conditioned authority schedules with pathway-specific diagnostics and explicit fallback policies. These design choices trade some peak performance for markedly higher stability under distribution shift.

An actionable evaluation protocol follows directly from this framing: test each channel under three modes—disabled, advisory, and authoritative—then report how both return and stability indicators move across modes. This protocol isolates whether gains come from signal quality or merely from aggressive authority. It also reveals hidden coupling effects, such as planners that are useful only when verifier thresholds are strict, or synthetic replay that helps only under conservative memory policies. These interaction tests are often more informative than incremental benchmark gains and should become standard in comparative studies.

Coupled-channel cascades are a particularly underreported failure class. A representative pattern is "planner drift $\rightarrow$ verifier over-trust $\rightarrow$ synthetic amplification": small feasibility errors in planner proposals increase reliance on verifier gating and on synthetic replay to patch coverage gaps, which can temporarily stabilize return while silently increasing bias. When verifier calibration degrades under shift, the system begins admitting low-quality trajectories; if these are then used to seed synthetic rollouts or to update persistent memory, the bias becomes self-reinforcing and collapse can be abrupt rather than gradual. The key design implication is that mitigation knobs must be coordinated: reducing planner authority without contracting verifier influence (or without tightening synthetic quotas) often fails because the remaining channels simply absorb the authority mass. Cross-channel rollback triggers

 doi:10.20944/preprints202603.0229.v1

based on disagreement spikes and divergence growth are therefore more effective than single-channel thresholding [1,5,8,16,19,20].

Another common cascade is "tool-memory contamination → planner context drift → verifier miscalibration". A transient tool outage or stale retrieval can be written into memory and then reused as if it were a trustworthy state variable. Because planning and verification both condition on context, the same corrupted fragment can bias subgoal proposals and critique scores across many decision steps, even when environment feedback is weak or delayed. Mitigations include typed memory writes with provenance, periodic re-grounding against fresh tool observations, and quarantine policies that freeze memory updates when tool reliability drops or contradictions are detected [5,12–14,19].

Ethics and Safety Safeguards

Authority escalation should be treated as a safety-critical decision rather than a pure performance optimization. In practice, high-authority channels benefit from guardrails such as capped influence, disagreement-triggered rollback, and explicit human-in-the-loop override policies in deployment-facing settings [51,59,82]. Failure containment should also be pathway-specific: planner faults require fast feasibility fallback, verifier faults require trust contraction, and tool-memory faults require context quarantine. These safeguards do not remove all risk, but they substantially reduce the probability that localized module errors become trajectory-wide failures.

From an optimization perspective, these safeguards can be interpreted as variance control under epistemic uncertainty: authority caps reduce heavy-tailed update noise, rollback policies prevent error persistence, and disagreement triggers limit correlated bias across channels. This interpretation links safety engineering directly to stable learning dynamics rather than treating it as a post hoc compliance layer [19,20,28].

Table 11. System-level failure modes, their mechanisms, and common mitigation patterns in LLM-enhanced RL.

Failure mode	Root cause	Observable symptom	Mitigation pattern	Representative papers
Planner-policy mismatch	Planner semantics exceed controller feasibility	High plan quality scores with low executed return	Feasibility critics, repair loops, soft authority rollback	[1,8,9]
Verifier exploitation	Proxy process scores weakly coupled to environment reward	Rising verifier confidence while task success plateaus	Disagreement ensembles, calibration sweeps, counterfactual reward checks	[19,20,24]
Synthetic replay drift	Generated trajectories outside target transition support	Value overestimation and brittle offline transfer	Confidence-weighted replay mixing, synthetic-real divergence penalties	[16,17,27]
Tool-memory contamination	Stale retrieval or malformed tool outputs persist in context state	Cascading action errors after early context faults	Provenance tags, typed writes, memory rollback policies	[5,12,13]
Compute-latency imbalance	Over-deliberation under fixed latency budgets	Marginal return gains with large runtime overhead	Uncertainty-triggered deliberation depth control	[33,47,88]

10. Research Agenda

The field has advanced quickly, but its scientific foundations remain incomplete. The principal challenge is not the absence of strong models; it is the absence of robust theories and protocols for mixed-supervision optimization. Language modules alter trajectory priors, supervision density, and dataset composition simultaneously, making causal attribution difficult. The following open problems identify research directions with direct implications for reliability.

10.1. Calibrated Verifier Uncertainty

Verifier scores are still frequently consumed as if they were calibrated probabilities, even when the underlying data regime is non-stationary. In practice, prompt variations, tool noise, and changing rollout composition can shift score semantics faster than threshold policies are updated. This mismatch turns ostensibly robust reweighting into brittle gating, especially in late training where policy

 doi:10.20944/preprints202603.0229.v1

behavior departs from verifier training data. A first priority is joint modeling of score and uncertainty, with explicit reliability intervals attached to each verification decision. A second priority is online recalibration against delayed environment outcomes so that verifier trust tracks real control quality rather than internal confidence alone [19,20,24].

10.2. Module Authority Arbitration

Most systems still assign module authority through fixed coefficients or static stage boundaries, even though module quality is strongly state- and phase-dependent. The same planner can be reliable in nominal regions and unreliable near boundary conditions where dynamics become nonlinear. Similarly, verifier confidence can lag policy drift, causing authority to remain high when it should contract. The research gap is not only better arbitration policies, but better arbitration objectives that optimize stability, not just short-horizon return. Promising directions include constrained meta-controllers that allocate authority under uncertainty budgets and can be evaluated independently from task reward [23,54,58].

10.3. Credit Assignment with Mixed Supervision

Theoretical understanding of policy updates under mixed supervision remains thin. Planner outputs, shaping terms, verifier weights, and synthetic replay are rarely independent; their errors are often correlated through shared context and model priors. When those correlations are positive, bias can accumulate faster than variance reduction benefits, producing deceptively stable but misaligned optimization. Future theory should therefore move beyond additive-noise assumptions and model cross-channel covariance explicitly. On the empirical side, standardized diagnostics should report not only channel-wise ablations but pairwise and higher-order interaction effects [6,21,65].

10.4. Uncertainty-Aware World Modeling

Generated rollouts improve coverage, but their utility depends on where and how they are inserted into training. Deep imagined trajectories can contribute useful structural priors while simultaneously amplifying small model errors into large value bias. A key open problem is uncertainty shaping across rollout depth: confidence should typically decay with horizon unless corroborated by grounded evidence. Another open problem is calibrating synthetic-real mixing adaptively, rather than fixing replay ratios globally. Robust pipelines should couple divergence diagnostics with automatic down-weighting when synthetic traces drift from observed transition statistics [15–18].

10.5. Planner-Controller Contracts

Planner-policy mismatch remains one of the fastest paths to catastrophic degradation in long-horizon tasks. High-level plans can be linguistically coherent yet violate actuation limits, temporal constraints, or hidden environmental preconditions. Contract-aware interfaces should therefore encode feasibility metadata directly in planner outputs, including confidence, precondition assumptions, and acceptable fallback behavior. Controllers, in turn, should return structured execution feedback that can trigger planner repair rather than silent failure accumulation. Closing this loop would transform planning from one-way suggestion to bidirectional coordination with measurable guarantees [1,6,8,9].

10.6. Tool and Memory Reliability

Tool and memory pathways are now central in interactive RL, yet reliability contracts remain underdeveloped. Retrieved or executed outputs are often inserted into context without explicit provenance, freshness, or confidence metadata. Once polluted context is written into memory, downstream planning and verification may repeatedly reuse it, creating long error cascades. Future work should treat memory governance as a first-class control problem with typed writes, provenance tracking, expiration policies, and rollback triggers. This framing also enables principled auditing of which context fragments causally influence policy updates [5,11–13].

 doi:10.20944/preprints202603.0229.v1

10.7. Synthetic Data Governance

Synthetic data has become indispensable in low-interaction regimes, but current governance is still largely ad hoc. Leakage between generated and evaluation distributions can silently inflate performance while masking weak generalization. Confounding can also occur when synthetic traces encode stylistic priors that correlate with benchmark scoring but not with robust control. The field needs auditable provenance standards that make generation source, filtering policy, and integration strength transparent. Influence-accounting tools that estimate how much performance depends on synthetic versus grounded data would make claims substantially more trustworthy [27,35,89,94].

10.8. Evaluation Standardization

Comparability across studies remains weak because critical protocol choices are inconsistently reported. Two papers with similar headline return may differ dramatically in prompt templates, tool permissions, memory truncation, verifier thresholds, and intervention schedules. Without this metadata, replication attempts often fail for reasons unrelated to algorithmic quality. Channel-aware benchmark cards can close this gap by standardizing reporting of authority policies, perturbation suites, and module-specific diagnostics. Beyond reporting, shared stress-test batteries are needed to evaluate robustness under prompt shift, tool failures, and delayed feedback perturbations [78,84,99,107].

10.9. Safety Under Adversarial Conditions

Adversarial pressure in modular RL systems is inherently cross-channel. Prompt manipulation can bias planner proposals, which then exploit verifier blind spots, while compromised tool outputs reinforce both channels through shared memory. Defenses designed for isolated modules are therefore insufficient when pathways interact. Safety architectures need coordinated detection across planner, verifier, simulator, and memory traces, with policies that reduce authority when multi-channel anomalies co-occur. Formal threat models for modular RL loops are still rare and should become a standard component of safety evaluation [51,59,82,123].

10.10. Scaling Laws for Modular RL Systems

Existing scaling analyses mostly track parameter count and total compute, but modular RL systems introduce a second scaling axis: orchestration topology. Performance may improve more from reallocating compute across planner, verifier, world, and policy channels than from increasing any single model size. This creates a resource-allocation problem that current scaling laws do not capture. Future work should model marginal return per compute unit for each channel under different task regimes, then optimize allocation jointly. Such module-aware scaling laws would enable predictable system design under fixed latency and cost constraints [45,47,53,60].

Table 12. Priority research problems, why they remain unresolved, and concrete forward directions.

Open problem	Why it is hard	Promising direction	Representative papers
Verifier calibration	Confidence-quality mismatch under shift	Uncertainty-aware verifier ensembles and adaptive weighting	[19,20,24]
Authority arbitration	State-dependent module reliability	Learned arbitration and uncertainty-triggered fallback	[23,54,57]
Mixed-supervision gradients	Correlated channel errors	Covariance-aware update estimators and bounds	[6,21,65]
World-model uncertainty	Hallucinated rollouts contaminate replay	Depth-aware confidence discounting and divergence checks	[16–18]
Planner-controller mismatch	Semantic plan vs dynamic feasibility gap	Contract-aware interfaces and online repair loops	[1,8,9]
Tool-memory reliability	Error persistence across long trajectories	Provenance-aware memory and rollback policies	[5,12,13]
Synthetic-data governance	Leakage and confounding in augmentation	Auditable data lineage and influence accounting	[27,35,94]
Evaluation standardization	Hidden protocol variables	Channel-aware benchmark cards and mandatory stress suites	[78,84,99]
Deployment safety	Cross-channel adversarial interactions	Runtime anomaly detection and safe authority reduction	[51,59,82]
Module-aware scaling	Compute allocation across heterogeneous channels	Joint scaling models for policy, verifier, planner, and simulator	[45,47,53]

 doi:10.20944/preprints202603.0229.v1

11. Conclusion

This survey examined how language-model modules improve reinforcement learning when embedded directly in the learning loop. The core claim is that these systems are best understood as supervision-pathway architectures, not as monolithic model substitutions. Information processors construct effective state, reward-verifier modules reshape optimization targets, decision-maker modules shape trajectory priors and search, and generator modules alter the data regime available to learning [1,4,5,16,33].

A mechanism-first view reveals both power and fragility. Gains are real in long-horizon, sparse-reward, and partially observable settings, especially when planner, verifier, and memory channels are coordinated. At the same time, each additional channel introduces new bias pathways and coupling risks. Reliable progress therefore depends on calibrated authority, uncertainty-aware arbitration, and pathway-level diagnostics [12,19,20,28].

The field now needs deeper theory and stronger experimental standards. Mixed-supervision gradient analysis, contract-aware module interfaces, auditable synthetic-data governance, and channel-aware benchmarks are central priorities. Work on these directions can shift the area from heuristic integration to principled optimization and make performance gains more durable under distribution shift [1,6,21,55,94,99].

From an engineering perspective, the most dependable systems will likely be those that treat language modules as probabilistic advisors with bounded authority, rather than immutable controllers. This framing supports practical safety mechanisms such as fallback policies, disagreement-triggered throttling, and incremental authority scaling. It also improves interpretability by making intervention pathways explicit and testable [5,19,20,51].

Ultimately, LLM-enhanced RL should be evaluated not only by peak benchmark scores but by supervision integrity: whether the pathway that produced a gain remains stable, auditable, and robust when environments, tools, and objectives evolve. A research program built around this criterion can turn current successes into dependable long-term capability [51,59,82].

Appendix A. Extended Coverage Citations

For compactness in the main text, full category-level citation bundles are provided below.

DM-A

[1,9–11,23,24,26,28,38–64].

RV-A

[20,65].

GN-A

[2–4,6–8,15–18,22,31,32,66–85].

DM-B

[19,21,34,86–88].

GN-B

[27,35,89–97].

IP-A

[5,12–14,25,29,30,98–110].

DM-C

[33,36,37,111–125].

 doi:10.20944/preprints202603.0229.v1

References

1. Ahn, M.J.; Brohan, A.; Brown, N.; Chebotar, Y.; Cortes, O.A.C.; David, B.; Finn, C.; Gopalakrishnan, K.; Hausman, K.; Herzog, A.; et al. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances, 2022, [2204.01691]. arXiv (Cornell University).
2. Liang, J.; Huang, W.; Xia, F.; Xu, P.; Hausman, K.; Ichter, B.; Florence, P.; Zeng, A. Code as Policies: Language Model Programs for Embodied Control, 2023. Unknown venue, https://doi.org/10.1109/icra48891.2023.10160591.
3. Wang, G.; Xie, Y.; Jiang, Y.; Mandlekar, A.; Xiao, C.; Zhu, Y.; Fan, L.; Anandkumar, A. Voyager: An Open-Ended Embodied Agent with Large Language Models, 2023, [2305.16291]. ArXiv.
4. Ma, Y.J.; Liang, W.; Wang, G.; Huang, D.A.; Bastani, O.; Jayaraman, D.; Zhu, Y.; Fan, L.; Anandkumar, A. Eureka: Human-Level Reward Design via Coding Large Language Models, 2023, [2310.12931]. ArXiv.
5. Lai, H.; Liu, X.; Iong, I.L.; Yao, S.; Chen, Y.; Shen, P.; Yu, H.; Zhang, H.; Zhang, X.; Dong, Y.; et al. AutoWebGLM: A Large Language Model-based Web Navigating Agent, 2024, [2404.03648]. ArXiv.
6. Peng, J.; Liu, Y.; Zhou, R.; Fleming, C.; Wang, Z.; Garcia, A.; Hong, M. HiPER: Hierarchical Reinforcement Learning with Explicit Credit Assignment for Large Language Model Agents, 2026, [2602.16165]. ArXiv.
7. Yuan, H.; Zhang, C.; Wang, H.; Xie, F.; Cai, P.; Dong, H.; Lu, Z. Skill Reinforcement Learning and Planning for Open-World Long-Horizon Tasks, 2023, [2303.16563]. ArXiv.
8. Dalal, M.; Chiruvolu, T.; Chaplot, D.; Salakhutdinov, R. Plan-Seq-Learn: Language Model Guided RL for Solving Long Horizon Robotics Tasks, 2024, [2405.01534]. ArXiv.
9. Liu, X.; Feng, J.; Zhuang, Z.; Zhao, J.; Que, M.; Li, J.; Wang, D.; Tong, H.; Chen, Y.; Li, P. CoDA: A Context-Decoupled Hierarchical Agent with Reinforcement Learning, 2025, [2512.12716]. ArXiv.
10. Xie, T.; Qi, Y.; Wen, J.; Wan, Z.; Dong, Y.; Wang, Z.; Cai, S.; Liang, Y.; Jia, T.; yuan wang.; et al. CREATE: Cross-Layer Resilience Characterization and Optimization for Efficient yet Reliable Embodied AI Systems, 2026, [2601.14140]. arXiv (Cornell University).
11. Cheng, M.; Ouyang, J.; Yu, S.; Yan, R.; Luo, Y.; Liu, Z.; Wang, D.; Liu, Q.; Chen, E. Agent-R1: Training Powerful LLM Agents with End-to-End Reinforcement Learning, 2025, [2511.14460]. ArXiv.
12. Li, Y.; Yi, M.; Li, X.; Fan, J.; Jiang, F.; Chen, B.; Li, P.; Song, J.; Zhang, T. Reasoning and Tool-use Compete in Agentic RL:From Quantifying Interference to Disentangled Tuning, 2026, [2602.00994]. ArXiv.
13. Xuan, W.; Zeng, Q.; Qi, H.; Xiao, Y.; Wang, J.; Yokoya, N. The Confidence Dichotomy: Analyzing and Mitigating Miscalibration in Tool-Use Agents, 2026, [2601.07264]. ArXiv.
14. Bai, H.; Taymanov, A.; Zhang, T.; Kumar, A.; Whitehead, S. WebGym: Scaling Training Environments for Visual Web Agents with Realistic Tasks, 2026, [2601.02439]. ArXiv.
15. Nottingham, K.; Ammanabrolu, P.; Suhr, A.; Choi, Y.; Hajishirzi, H.; Singh, S.; Fox, R. Do Embodied Agents Dream of Pixelated Sheep: Embodied Decision Making using Language Guided World Modelling, 2023, [2301.12050]. ArXiv.
16. Sharma, A.K.; Sun, Y.; Lu, N.; Zhang, Y.; Liu, J.; Yang, S. World-Gymnast: Training Robots with Reinforcement Learning in a World Model, 2026, [2602.02454]. ArXiv.
17. Yang, J.; Lin, K.; Li, J.; Zhang, W.; Lin, T.; Wu, L.; Su, Z.; Zhao, H.; Zhang, Y.Q.; Chen, L.; et al. RISE: Self-Improving Robot Policy with Compositional World Model, 2026, [2602.11075]. ArXiv.
18. Yang, S. World Models as an Intermediary between Agents and the Real World, 2026, [2602.00785]. ArXiv.
19. Song, Z.; Ma, Z.; Qiang, W.; Zheng, C.; Hua, G. Adaptive Uncertainty-Aware Tree Search for Robust Reasoning, 2026, [2602.06493]. ArXiv.
20. Cai, X.Q.; Sugiyama, M. VI-CuRL: Stabilizing Verifier-Independent RL Reasoning via Confidence-Guided Variance Reduction, 2026, [2602.12579]. ArXiv.
21. Guo, P.; Li, C.; Feng, Y.; Zhang, C. Code Evolution for Control: Synthesizing Policies via LLM-Driven Evolutionary Search, 2026, [2601.06845]. ArXiv.org.
22. Lü, Y.J.; Wang, C.; Shen, L.; Huang, J.; Xu, T. Mock Worlds, Real Skills: Building Small Agentic Language Models with Synthetic Tasks, Simulated Environments, and Rubric-Based Rewards, 2026, [2601.22511]. ArXiv.org.
23. Luo, X.; Zhang, Y.; He, Z.; Wang, Z.; Zhao, S.; Li, D.; Qiu, L.K.; Yang, Y. Agent Lightning: Train ANY AI Agents with Reinforcement Learning, 2025, [2508.03680]. ArXiv.
24. Nellessen, S.; Kachman, T. David vs. Goliath: Verifiable Agent-to-Agent Jailbreaking via Reinforcement Learning, 2026, [2602.02395]. ArXiv.
25. Ye, R.; Zhang, Z.; Li, K.; Yin, H.; Tao, Z.; Zhao, Y.; Su, L.; Zhang, L.; Qiao, Z.; Wang, X.; et al. AgentFold: Long-Horizon Web Agents with Proactive Context Management, 2025, [2510.24699]. ArXiv.

 doi:10.20944/preprints202603.0229.v1

26. Vijay, D.; Ethiraj, V. Graph-Symbolic Policy Enforcement and Control (G-SPEC): A Neuro-Symbolic Framework for Safe Agentic AI in 5G Autonomous Networks, 2025, [2512.20275]. ArXiv.
27. Gu, C.; Pan, Y.; Xiong, H.; Chen, Y. STO-RL: Offline RL under Sparse Rewards via LLM-Guided Subgoal Temporal Order, 2026, [2601.08107]. ArXiv.org.
28. Liu, C.; Zhao, Y.; Liu, L.; Ye, Y.; Szepesvári, C.; Yang, L.F. LACONIC: Length-Aware Constrained Reinforcement Learning for LLM, 2026, [2602.14468]. ArXiv.
29. Ding, H.; Liu, P.; Wang, J.; Ji, Z.; Cao, M.; Zhang, R.; Ai, L.; Yang, E.; Shi, T.; Yu, L. DynaWeb: Model-Based Reinforcement Learning of Web Agents, 2026, [2601.22149]. ArXiv.
30. Xie, T.; Zhao, S.; Wu, C.; Liu, Y.; Luo, Q.; Zhong, V.W.; Yang, Y.; Yu, C. Text2Reward: Reward Shaping with Language Models for Reinforcement Learning, 2023, [2309.11489]. arXiv (Cornell University).
31. Song, J.; Zhou, Z.; Liu, J.; Fang, C.; Shu, Z.; Ma, L. Self-Refined Large Language Model as Automated Reward Function Designer for Deep Reinforcement Learning in Robotics, 2023, [2309.06687]. arXiv (Cornell University).
32. Team, G.; Wang, B.; Ni, C.; Huang, G.; Zhao, G.; Li, H.; Li, J.; Lv, J.; Liu, J.; Feng, L.; et al. GigaBrain-0.5M*: a VLA That Learns From World Model-Based Reinforcement Learning, 2026, [2602.12099]. ArXiv.
33. Wang, S.; Li, P.; Fu, Y.; Liu, K.; Li, F.; Liu, Y.; Sun, X.; Li, Z.; Zhao, S.; Zhao, J.; et al. MARTI-MARS^2: Scaling Multi-Agent Self-Search via Reinforcement Learning for Code Generation, 2026, [2602.07848]. ArXiv.
34. for Artificial Intelligence 2026, A.; Duan, Y.; Jian, L.; Shi, Y. Navigating the Alpha Jungle: An LLM-Powered MCTS Framework for Formulaic Alpha Factor Mining, 2026. Underline Science Inc., https://doi.org/10.48448/q4pk-2g71.
35. Shi, R.; Liu, Y.; Ze, Y.; Du, S.S.; Xu, H. Unleashing the Power of Pre-trained Language Models for Offline Reinforcement Learning, 2023, [2310.20587]. ArXiv.
36. Hu, H.; Sadigh, D. Language Instructed Reinforcement Learning for Human-AI Coordination, 2023, [2304.07297]. ArXiv.
37. Li, Y.; Zhang, X.; Lu, W.; Tang, Z.; Wu, M.; Luo, H.; Wu, T.; Peng, Z.; Mi, H.; Feng, Y.; et al. Who Deserves the Reward? SHARP: Shapley Credit-based Optimization for Multi-Agent System, 2026, [2602.08335]. ArXiv.
38. Zheng, Q.; Zhang, A.; Grover, A. Online Decision Transformer, 2022, [2202.05607]. arXiv (Cornell University).
39. Yeo, E.; Tong, Y.; Niu, M.; Neubig, G.; Yue, X. Demystifying Long Chain-of-Thought Reasoning in LLMs, 2025, [2502.03373]. ArXiv.
40. Christakopoulou, K.; Qu, I.; Canny, J.; Goodridge, A.; Adams, C.; Chen, M.; Matarić, M. Conversational Planning for Personal Plans, 2025, [2502.19500]. ArXiv.
41. Chen, Y.; Liu, Y.; Zhou, J.; Hao, Y.; Wang, J.; Zhang, Y.; Li, N.; Fan, C. R1-Code-Interpreter: LLMs Reason with Code via Supervised and Multi-stage Reinforcement Learning, 2025, [2505.21668]. ArXiv.
42. Gong, M.; Huang, X.; Xu, Z.; Asari, V.K. MindFlow+: A Self-Evolving Agent for E-Commerce Customer Service, 2025, [2507.18884]. ArXiv.
43. Dong, G.; Mao, H.; Ma, K.; Bao, L.; Chen, Y.; Wang, Z.; Chen, Z.; Du, J.; Wang, H.; Zhang, F.; et al. Agentic Reinforced Policy Optimization, 2025, [2507.19849]. ArXiv.
44. Ding, H.; Huang, B.; Fang, Y.; Liao, W.; Jiang, X.; Li, Z.; Zhao, J.; Wang, Y. ProMed: Shapley Information Gain Guided Reinforcement Learning for Proactive Medical LLMs, 2025, [2508.13514]. ArXiv.
45. Zhang, G.; Geng, H.; Yu, X.; Yin, Z.; Zhang, Z.; Tan, Z.; Zhou, H.; Li, Z.; Xue, X.; Li, Y.; et al. The Landscape of Agentic Reinforcement Learning for LLMs: A Survey, 2025, [2509.02547]. ArXiv.
46. Zhao, X.; Yan, M.; Zhang, Y.; Deng, Y.; Wang, J.; Zhu, F.; Qiu, Y.; Cheng, H.; Chua, T.S. Reinforced Strategy Optimization for Conversational Recommender Systems via Network-of-Experts, 2025, [2509.26093]. ArXiv.
47. Lu, M.; Sun, W.; Du, W.; Ling, Z.; Yao, X.; Liu, K.; Chen, J. Scaling LLM Multi-turn RL with End-to-end Summarization-based Context Management, 2025, [2510.06727]. ArXiv.
48. Zhang, Y.; Xie, L.; Qiu, R.; Liu, J.; Wang, S. Beyond Static LLM Policies: Imitation-Enhanced Reinforcement Learning for Recommendation, 2025, [2510.13229]. ArXiv.
49. Wang, G.; Dai, S.; Ye, G.; Gan, Z.; Yao, W.; Deng, Y.; Wu, X.; Ying, Z. Information Gain-based Policy Optimization: A Simple and Effective Approach for Multi-Turn LLM Agents, 2025, [2510.14967]. ArXiv.
50. Peng, C.; Zhang, Z.; Chi, C.; Wei, X.; Zhang, Y.; Wang, H.; Wang, P.; Wang, Z.; Liu, J.; Zhang, S. PIGEON: VLM-Driven Object Navigation via Points of Interest Selection, 2025, [2511.13207]. arXiv (Cornell University).
51. Shu, Y.; Xu, J.; Tang, J.; Gao, R.L.; Sun, C. LA-RL: Language Action-guided Reinforcement Learning with Safety Guarantees for Autonomous Highway Driving, 2025, [2512.05686]. arXiv (Cornell University).
52. Jiang, Y.; Jiang, L.; Teney, D.; Moor, M.; Brbic, M. Meta-RL Induces Exploration in Language Agents, 2025, [2512.16848]. ArXiv.

 doi:10.20944/preprints202603.0229.v1

53. Kar, I.; Zonunpuia, S.; Ralte, Z. Towards AGI A Pragmatic Approach Towards Self Evolving Agent, 2026, [2601.11658]. ArXiv.
54. Ma, Y.; Li, L.; chen, Y.; Li, P.; Li, X.; Guo, Q.; Lin, D.; Chen, K. Timely Machine: Awareness of Time Makes Test-Time Scaling Agentic, 2026, [2601.16486]. ArXiv.
55. Lei, Z.; Liu, G.; Zhang, Y.; Liu, Q.; Wen, C.; Zhang, S.; Lian, W.; Chen, S. EmboCoach-Bench: Benchmarking AI Agents on Developing Embodied Robots, 2026, [2601.21570]. ArXiv.
56. Yoon, S.H.; Qian, R.; Zhao, M.; Li, W.; Wang, M. TrailBlazer: History-Guided Reinforcement Learning for Black-Box LLM Jailbreaking, 2026, [2602.06440]. ArXiv.
57. Hu, T.; Fu, Q.; Chen, Y.; Liu, Z.; Ding, B. SeeUPO: Sequence-Level Agentic-RL with Convergence Guarantees, 2026, [2602.06554]. ArXiv.
58. Cai, Y.; Liu, Z.; Zhu, X.; Wang, C.; Chen, J.; Wang, H.; Wang, W.C.; Jin, D.; Chen, S. AceGRPO: Adaptive Curriculum Enhanced Group Relative Policy Optimization for Autonomous Machine Learning Engineering, 2026, [2602.07906]. ArXiv.
59. Yang, J.; Hou, J.; Yu, X.; He, W.; Wu, Y. CAPER: Constrained and Procedural Reasoning for Robotic Scientific Experiments, 2026, [2602.09367]. ArXiv.
60. Wang, B.; Li, K.; Liu, T.; Li, C.; Wang, J.; Zhang, Y.; Cheng, J. LLM-Based Scientific Equation Discovery via Physics-Informed Token-Regularized Policy Optimization, 2026, [2602.10576]. ArXiv.
61. Huang, S.; Cohn, T.; Lipovetzky, N. Chasing Progress, Not Perfection: Revisiting Strategies for End-to-End LLM Plan Generation, 2025. Proceedings of the International Conference on Automated Planning and Scheduling, https://doi.org/10.1609/icaps.v35i1.36119.
62. Gu, N.; Hayashibe, M.; Kutsuzawa, K.; Yu, H. Deep learning-based robotic cloth manipulation applications: systematic review, challenges and opportunities for physical AI, 2026. Frontiers in Robotics and AI, https://doi.org/10.3389/frobt.2026.1752914.
63. Pesjak, D.; Žabkar, J. Robot Planning via LLM Proposals and Symbolic Verification, 2026. Machine Learning and Knowledge Extraction, https://doi.org/10.3390/make8010022.
64. Müller-Brockhausen.; Matthias. Exploring the synergies between transfer in reinforcement learning and procedural content generation, 2025. Unknown venue, https://doi.org/10.60602/1887/4282228.
65. Fan, J.; Ren, R.; Li, J.; Pandey, R.; Shivakumar, P.G.; Bulyko, I.; Gandhe, A.; Liu, G.; Gu, Y. Incentivizing Consistent, Effective and Scalable Reasoning Capability in Audio LLMs via Reasoning Process Rewards, 2025, [2510.20867]. ArXiv.
66. Paischer, F.; Adler, T.; Patil, V.; Bitto-Nemling, A.; Holzleitner, M.; Lehner, S.; Eghbal-zadeh, H.; Hochreiter, S. History Compression via Language Models in Reinforcement Learning, 2022, [2205.12258]. arXiv (Cornell University).
67. Du, Y.; Watkins, O.; Wang, Z.; Colas, C.; Darrell, T.; Abbeel, P.; Gupta, A.; Andreas, J. Guiding Pretraining in Reinforcement Learning with Large Language Models, 2023, [2302.06692]. ArXiv.
68. Kwon, M.; Xie, S.M.; Bullard, K.; Sadigh, D. Reward Design with Language Models, 2023, [2303.00001]. arXiv (Cornell University).
69. Akyürek, A.F.; Akyürek, E.; Madaan, A.; Kalyan, A.; Clark, P.; Wijaya, D.; Tandon, N. RL4F: Generating Natural Language Feedback with Reinforcement Learning for Repairing Model Outputs, 2023, [2305.08844]. ArXiv.
70. Szot, A.; Schwarzer, M.; Agrawal, H.; Mazoure, B.; Talbott, W.; Metcalf, K.; Mackraz, N.; Hjelm, D.; Toshev, A. Large Language Models as Generalizable Policies for Embodied Tasks, 2023, [2310.17722]. ArXiv.
71. Deng, Y.; Zhang, W.; Lam, W.; Ng, S.K.; Chua, T.S. Plug-and-Play Policy Planner for Large Language Model Powered Dialogue Agents, 2023, [2311.00262]. ArXiv.
72. Zhou, Z.; Hu, B.; Zhao, C.; Zhang, P.; Liu, B. Large Language Model as a Policy Teacher for Training Reinforcement Learning Agents, 2023, [2311.13373]. ArXiv.
73. Cheng, Y.; Zhang, C.; Zhang, Z.; Meng, X.; Hong, S.; Li, W.; Wang, Z.; Wang, Z.; Yin, F.; Zhao, J.; et al. Exploring Large Language Model based Intelligent Agents: Definitions, Methods, and Prospects, 2024, [2401.03428]. ArXiv.
74. Zhou, Y.; Zanette, A.; Pan, J.; Levine, S.; Kumar, A. ArCHer: Training Language Model Agents via Hierarchical Multi-Turn RL, 2024, [2402.19446]. ArXiv.
75. Chen, Y.; han Ding, Z.; Wang, Z.; Wang, Y.; Zhang, L.; Liu, S. Asynchronous Large Language Model Enhanced Planner for Autonomous Driving, 2024, [2406.14556]. ArXiv.
76. Yang, Z.; Shen, W.; Li, C.; Chen, R.; Wan, F.; Yan, M.; Quan, X.; Huang, F. SPELL: Self-Play Reinforcement Learning for Evolving Long-Context Language Models, 2025, [2509.23863]. ArXiv.

 doi:10.20944/preprints202603.0229.v1

77. Lu, S.; Wang, Z.; Zhang, H.; Wu, Q.; Gan, L.; Zhuang, C.; Gu, J.; Lin, T. Don't Just Fine-tune the Agent, Tune the Environment, 2025, [2510.10197]. ArXiv.
78. Sang, J.; Xiao, J.; Han, J.; Chen, J.; Chen, X.; Wei, S.; Sun, Y.; Wang, Y. Beyond Pipelines: A Survey of the Paradigm Shift toward Model-Native Agentic AI, 2025, [2510.16720]. ArXiv.
79. Zhang, J.; Cui, W.; Li, Z.; Huang, L.; Malin, B.; Xiong, C.; Wu, C.S. From Passive Metric to Active Signal: The Evolving Role of Uncertainty Quantification in Large Language Models, 2026, [2601.15690]. ArXiv.
80. Wang, P.; Wu, Y.; Song, X.; Wang, W.; Chen, G.; Li, Z.; Yan, K.; Deng, K.; Liu, Q.; Zhao, S.; et al. ShopSimulator: Evaluating and Exploring RL-Driven LLM Agent for Shopping Assistants, 2026, [2601.18225]. ArXiv.
81. Ni, J.; Pu, J.; Yang, Z.; Luo, J.; Hu, C. STAR: Similarity-guided Teacher-Assisted Refinement for Super-Tiny Function Calling Models, 2026, [2602.03022]. ArXiv.
82. Gao, Y.; Hammar, K.; Li, T. In-Context Autonomous Network Incident Response: An End-to-End Large Language Model Agent Approach, 2026, [2602.13156]. ArXiv.
83. Pallagani, V.; Muppasani, B.; Roy, K.; Fabiano, F.; Loreggia, A.; Murugesan, K.; Srivastava, B.; Rossi, F.; Horesh, L.; Sheth, A. On the Prospects of Incorporating Large Language Models (LLMs) in Automated Planning and Scheduling (APS), 2024. Proceedings of the International Conference on Automated Planning and Scheduling, https://doi.org/10.1609/icaps.v34i1.31503.
84. Jeong, H.; Lee, H.; Kim, C.; Shin, S. A Survey of Robot Intelligence with Large Language Models, 2024. Applied Sciences, https://doi.org/10.3390/app14198868.
85. for Artificial Intelligence 2026, A.; Gurevych, I.; Rizvi, M.I.H.; Zhu, X. SPARE: Single-Pass Annotation with Reference-Guided Evaluation for Automatic Process Supervision and Reward Modelling, 2026. Underline Science Inc., https://doi.org/10.48448/0js4-fm43.
86. Zhao, Y.; Huang, W.; Wang, S.; Zhao, R.; Chen, C.; Shu, Y.; Qin, C. Training Multi-Turn Search Agent via Contrastive Dynamic Branch Sampling, 2026, [2602.03719]. ArXiv.
87. Zhong, W.; Yang, J.; Wu, Y.; Liu, Y.; Yao, J.; Kuang, K. SIGHT: Reinforcement Learning with Self-Evidence and Information-Gain Diverse Branching for Search Agent, 2026, [2602.11551]. ArXiv.
88. Chu, Z.; Wang, X.; Hong, J.; Fan, H.; Huang, Y.; Yang, Y.; Xu, G.; Zhao, C.; Xiang, C.; Hu, S.; et al. REDSearcher: A Scalable and Cost-Efficient Framework for Long-Horizon Search Agents, 2026, [2602.14234]. ArXiv.
89. Hong, J.; Dragan, A.; Levine, S. Planning without Search: Refining Frontier LLMs with Offline Goal-Conditioned RL, 2025, [2505.18098]. ArXiv.
90. Hou, C.; Wu, K.; Liu, J.; Che, Z.; Wu, D.; Liao, F.; Li, G.; He, J.; Feng, Q.; Jin, Z.; et al. RoboMIND 2.0: A Multimodal, Bimanual Mobile Manipulation Dataset for Generalizable Embodied Intelligence, 2025, [2512.24653]. ArXiv.org.
91. Liu, Y.; Liu, Y.; Li, Z.; Huang, Y.; Feng, X.; Hu, Z.; Hu, J.; Yan, J.; Lian, F.; Liu, Y. UltraLogic: Enhancing LLM Reasoning through Large-Scale Data Synthesis and Bipolar Float Reward, 2026, [2601.03205]. ArXiv.
92. Kumar, P.; Saran, V.; Patel, D.; Kulkarni, N.; Vereshchaka, A. Attention-Based Offline Reinforcement Learning and Clustering for Interpretable Sepsis Treatment, 2026, [2601.14228]. ArXiv.org.
93. Hübotter, J.; Lübeck, F.; Behric, L.; Baumann, A.; Bagatella, M.; Marta, D.; Hakimi, I.; Shenfeld, I.; Buening, T.K.; Guestrin, C.; et al. Reinforcement Learning via Self-Distillation, 2026, [2601.20802]. ArXiv.
94. Li, M.; Wang, R.; Tan, R.; Wen, Y. DCoPilot: Generative AI-Empowered Policy Adaptation for Dynamic Data Center Operations, 2026, [2602.02137]. ArXiv.
95. Khan, R.M.S.; Liu, Z.; Tan, Z.; Fleming, C.; Chen, T. TMS: Trajectory-Mixed Supervision for Reward-Free, On-Policy SFT, 2026, [2602.03073]. ArXiv.
96. Xu, Q.; Habib, G.; Wu, F.; Du, Y.; Chen, Z.; Mishra, S.; Perera, D.; Feng, M. medR: Reward Engineering for Clinical Offline Reinforcement Learning via Tri-Drive Potential Functions, 2026, [2602.03305]. ArXiv.
97. Amer-Yahia, S. Intelligent Agents for Data Exploration, 2024. Proceedings of the VLDB Endowment, https://doi.org/10.14778/3685800.3685913.
98. Furuta, H.; Lee, K.H.; Nachum, O.; Matsuo, Y.; Faust, A.; Gu, S.S.; Gur, I. Multimodal Web Navigation with Instruction-Finetuned Foundation Models, 2023, [2305.11854]. ArXiv.
99. Qi, Z.; Liu, X.; Iong, I.L.; Lai, H.; Sun, X.; Zhao, W.; Yang, Y.; Yang, X.; Sun, J.; Yao, S.; et al. WebRL: Training LLM Web Agents via Self-Evolving Online Curriculum Reinforcement Learning, 2024, [2411.02337]. ArXiv.
100. Zhuang, Y.; Jin, D.; Chen, J.; Shi, W.; Wang, H.; Zhang, C. WorkForceAgent-R1: Incentivizing Reasoning Capability in LLM-based Web Agents via Reinforcement Learning, 2025, [2505.22942]. ArXiv.
101. Liu, J.; Li, Y.; Zhang, C.; Li, J.; Chen, A.; Ji, K.; Cheng, W.; Wu, Z.; Du, C.; Xu, Q.; et al. WebExplorer: Explore and Evolve for Training Long-Horizon Web Agents, 2025, [2509.06501]. ArXiv.

 doi:10.20944/preprints202603.0229.v1

102. Zhang, Y.; Zeng, Y.; Li, Q.; Hu, Z.; Han, K.; Zuo, W. Tool-R1: Sample-Efficient Reinforcement Learning for Agentic Tool Use, 2025, [2509.12867]. ArXiv.
103. Tan, W.; Qu, X.; Tu, M.; Ge, M.; Liu, A.T.; Koehn, P.; Lu, L. Process-Supervised Reinforcement Learning for Interactive Multimodal Tool-Use Agents, 2025, [2509.14480]. ArXiv.
104. Choi, C.; Song, H.; Kim, D.; Jung, W.; Cho, M.; Park, S.; Bae, N.; Yu, S.; Lim, K. MENTOR: A Reinforcement Learning Framework for Enabling Tool Use in Small Models via Teacher-Optimized Rewards, 2025, [2510.18383]. ArXiv.
105. Wu, J.; Zhao, Q.; Chen, Z.; Qin, K.; Zhao, Y.; Wang, X.; Yao, Y. GAP: Graph-Based Agent Planning with Parallel Tool Use and Reinforcement Learning, 2025, [2510.25320]. ArXiv.
106. Jiang, Y.; Ferraro, F. SCRIBE: Structured Mid-Level Supervision for Tool-Using Language Models, 2026, [2601.03555]. arXiv (Cornell University).
107. Gao, X.; Huang, P.; Liu, Z.; Yan, Y.; Wang, S.; Chen, Z.; Qian, C.; Yu, G.; Gu, Y. Teaching LLMs to Learn Tool Trialing and Execution through Environment Interaction, 2026, [2601.12762]. ArXiv.
108. Lin, M.; Dai, E.; Liu, H.; Tang, X.; Yan, Y.; Dai, Z.; Zeng, J.; Zhang, Z.; Wang, F.; Gao, H.; et al. How Far Are LLMs from Professional Poker Players? Revisiting Game-Theoretic Reasoning with Agentic Tool Use, 2026, [2602.00528]. ArXiv.
109. Guo, Y.; Yang, W.; Yang, S.; Liu, Z.; Chen, C.; Wei, Y.; Hu, Y.; Huang, Y.; Hao, G.; Yuan, D.; et al. OpAgent: Operator Agent for Web Navigation, 2026, [2602.13559]. ArXiv.
110. Shen, J.H.; Jain, A.; Xiao, Z.; Amlekar, I.; Hadji, M.; Podolny, A.; Talwalkar, A. ScribeAgent: Towards Specialized Web Agents Using Production-Scale Workflow Data, 2024. Unknown venue, https://doi.org/10.32388/8vog0o.
111. Kannan, S.S.; Venkatesh, V.L.N.; Min, B.C. SMART-LLM: Smart Multi-Agent Robot Task Planning using Large Language Models, 2023, [2309.10062]. ArXiv.
112. Agashe, S.; Fan, Y.; Reyna, A.; Wang, X.E. LLM-Coordination: Evaluating and Analyzing Multi-agent Coordination Abilities in Large Language Models, 2023, [2310.03903]. ArXiv.
113. Hu, M.; Zhou, Y.; Fan, W.; Nie, Y.; Xia, B.; Sun, T.; Ye, Z.; Jin, Z.; Li, Y.; Chen, Q.; et al. OWL: Optimized Workforce Learning for General Multi-Agent Assistance in Real-World Task Automation, 2025, [2505.23885]. ArXiv.
114. Li, W.; Lin, J.; Jiang, Z.; Cao, J.; Liu, X.; Zhang, J.; Huang, Z.; Chen, Q.; Sun, W.; Wang, Q.; et al. Chain-of-Agents: End-to-End Agent Foundation Models via Multi-Agent Distillation and Agentic RL, 2025, [2508.13167]. ArXiv.
115. Zhou, Y.; Zhang, M.; Li, K.; Wang, M.; Liu, Q.; Wang, Q.; Liu, J.; Liu, F.; Li, S.; Li, W.; et al. Mixture-of-Minds: Multi-Agent Reinforcement Learning for Table Understanding, 2025, [2510.20176]. ArXiv.
116. Ali, R.S.; Beltran-Hernandez, C.C.; Wan, W.; Harada, K. Learning-based Cooperative Robotic Paper Wrapping: A Unified Control Policy with Residual Force Control, 2025, [2511.03181]. arXiv (Cornell University).
117. Liu, S.; Du, D.; Yang, T.; Li, Y.; Qiu, B. MarsRL: Advancing Multi-Agent Reasoning System via Reinforcement Learning with Agentic Pipeline Parallelism, 2025, [2511.11373]. ArXiv.
118. Sharma, A.; Arafat, S.Y.; Sharma, J.K.; Huang, K. Bilevel Optimization for Covert Memory Tampering in Heterogeneous Multi-Agent Architectures (XAMT), 2025, [2512.15790]. ArXiv.
119. Zhou, Z.; Liu, Z.; Liu, J.; Shao, Q.; Wang, Y.; Shao, K.; Jin, D.; Xu, F. ResMAS: Resilience Optimization in LLM-based Multi-agent Systems, 2026, [2601.04694]. ArXiv.
120. Zou, M.; Chen, J.; Luo, A.; Dai, J.; Zhang, C.; Sun, D.; Xu, Z. FinEvo: From Isolated Backtests to Ecological Market Games for Multi-Agent Financial Strategy Evolution, 2026, [2602.00948]. ArXiv.
121. Xu, Z.; Xu, Z.; Zhang, R.; Zhu, C.; Yu, S.; Liu, W.; Zhang, Q.; Ding, W.; Yu, C.; Wang, Y. WideSeek-R1: Exploring Width Scaling for Broad Information Seeking via Multi-Agent Reinforcement Learning, 2026, [2602.04634]. ArXiv.
122. Despature, J.Y.; Shibata, K.; Matsubara, T. CoLF: Learning Consistent Leader-Follower Policies for Vision-Language-Guided Multi-Robot Cooperative Transport, 2026, [2602.07776]. ArXiv.
123. Chen, J.; Yang, H.; Liu, Z.; Joe-Wong, C. The Five Ws of Multi-Agent Communication: Who Talks to Whom, When, What, and Why – A Survey from MARL to Emergent Language and LLMs, 2026, [2602.11583]. ArXiv.

 doi:10.20944/preprints202603.0229.v1

124. Li, H.; Yu, C.; Stepputtis, S.; Campbell, J.; Hughes, D.; Lewis, C.; Sycara, K. Theory of Mind for Multi-Agent Collaboration via Large Language Models, 2023. Unknown venue, https://doi.org/10.18653/v1/2023.emnlp-main.13.
125. for Artificial Intelligence 2026, A.; Ma, Q.; Xie, X.; Xue, X.; Zhang, X. Adaptive Regulation via Dual-Layer Evolution (ARDE): A Multi-Agent Approach to Balancing Efficiency, Fairness, and Diversity in Crowdsourced Platforms, 2026. Underline Science Inc., https://doi.org/10.48448/8p81-ba98.