The Harness Thesis

Reliability is the next Frontier

January 2026

White Paper

Author: Karl Doose

Contents

Executive Summary

In late 2025, a six-person startup achieved state-of-the-art on ARC-AGI-2, the most demanding test of machine reasoning yet devised. They beat Google at half the cost. They trained no models. They built a better harness.[1]

This paper argues that AI capability scales along two axes: Model Scale and Harness Sophistication. The industry has invested hundreds of billions in the first while leaving the second largely unexplored. We propose that harness sophistication now offers greater returns, and we explain why.

The core problem is verification. Foundation models have become fluent without becoming reliably correct. Even impressive per-step accuracy compounds into failure across sequential decisions: a model achieving 95% accuracy on individual steps yields only 36% success probability across twenty steps. Self-verification cannot close this gap. When a model evaluates its own output, the verifier shares the generator's blind spots.

We present a harness architecture with three components: an Orchestrator governing iteration and stopping, a Preference Learning System capturing judgment without modifying model weights, and a Verification Layer spanning formal provers to simulation-based world models. We ground this architecture in a worked example and identify the conditions under which each component delivers value.

The paper's central theoretical contribution is the Convergence Thesis: robust verification of complex claims requires the same capabilities as generating those claims reliably. The verification gap and the AGI gap are the same gap.

This paper is for researchers working on verification, alignment, or world models, investors assessing where value will accumulate in the AI stack, and enterprise leaders evaluating AI deployment.

Part I: Why Impact Will Stall

Chapter 1: Two Scaling Axes

Reframing AI Progress

The artificial intelligence industry has spent years climbing a single axis. Since the publication of the original scaling laws papers, the dominant strategy for improving AI capabilities has followed a straightforward formula: train larger models on more data with more compute. This approach delivered GPT-3, then GPT-4, then a succession of frontier models that progressively expanded what machines could do. The strategy worked. Each generation brought capabilities that seemed impossible from the generation before.

But the returns have changed. By late 2025, training costs for frontier models exceed $100 million per run, with projections suggesting billion-dollar training runs within two years. The gains from each dollar spent have compressed. Models have become remarkably fluent without becoming reliably correct, generating impressive outputs that still require careful human verification. The need for that verification has not diminished as models have grown. Something in the dominant strategy has stopped working, or perhaps never worked the way the industry assumed.

The Poetiq result suggests what that something might be. In November 2025, the AI research company achieved state-of-the-art performance on ARC-AGI-2, a benchmark designed to test genuine reasoning through abstract visual puzzles requiring symbolic interpretation and compositional reasoning. Leading AI models had scored under 5% on this benchmark in early 2025; Poetiq's system reached 54% on the verified semi-private evaluation. The previous best, Gemini 3 Deep Think, had achieved 45% at a cost of $77 per problem. Poetiq reached higher accuracy at $31 per problem. They did not train a better model. They built a better harness.

The system worked by wrapping an unchanged foundation model in sophisticated iteration and verification loops. It would generate a candidate solution, test it against the puzzle's example cases, analyze why it failed, and generate an improved candidate. This cycle repeated until the solution worked or computational budget ran out. The same model, with the same weights and the same training, performed dramatically better when embedded in infrastructure designed to verify and refine its outputs. The ARC Prize organization characterized 2025 as the year of the

refinement loop, noting that progress is now being driven by systems that verify and refine at the application layer rather than by scaling model size alone.[5]

This result illuminates a structural feature of AI capability that the industry's dominant narrative has obscured. Capability scales along two independent axes, not one. The first axis is model scale: the raw capability encoded in foundation model weights through pre-training. Larger models encode more world knowledge, more linguistic patterns, more reasoning heuristics. They perform better on a first attempt. The industry has optimized this axis relentlessly, and the optimization has delivered real gains. But first attempts are often wrong, and model scale does not address what happens after the first attempt fails.

The second axis is harness sophistication: the intelligence of systems that wrap the foundation model. This includes verification layers that check outputs against external criteria, refinement loops that iterate toward correct solutions, and preference systems that learn what specific users consider good. A sophisticated harness extracts more value from a fixed model by catching errors, directing iteration, and accumulating feedback. The Poetiq result demonstrates that returns to harness sophistication can exceed returns to model scale, at least for tasks where verification is possible. Their system achieved higher accuracy at lower cost not by improving the model but by improving everything around it.

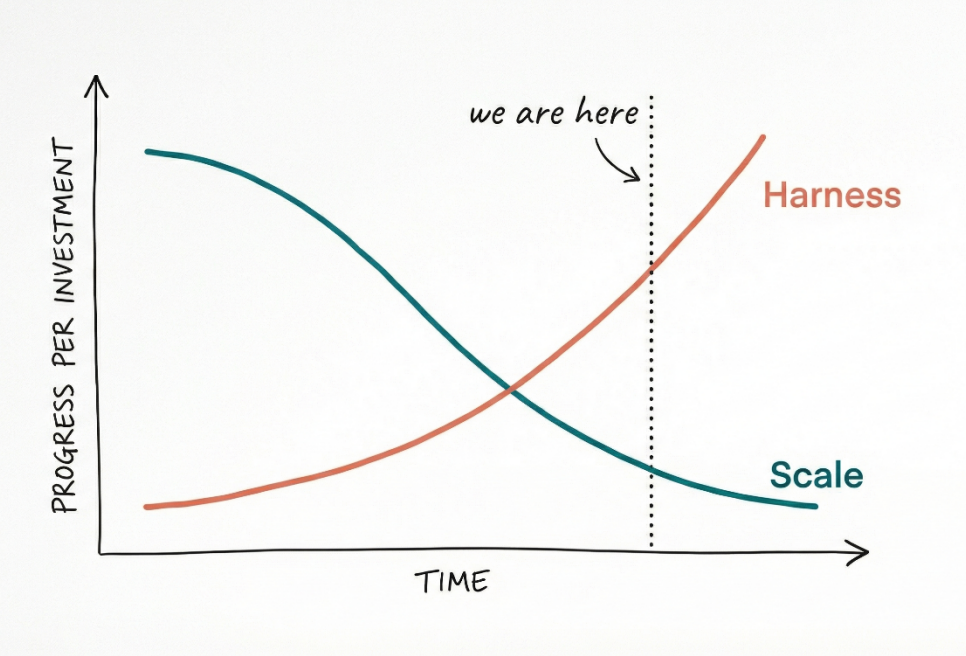

Figure 1: Inversion of Harness and Scale ROI over time

The two axes are not substitutes. A sophisticated harness cannot compensate for a model that lacks basic competence; you cannot iterate toward a solution if the model cannot generate plausible candidates. But the axes are also not redundant. A powerful model without verification remains unreliable, its errors simply more confident and harder to detect. The strategic implications run deep: if harness sophistication offers greater returns than model scale for practical applications, then competitive advantage shifts from training capability to orchestration capability. The organizations that thrive will not necessarily be those that train the largest models. They will be those that build the most effective systems for verifying, refining, and personalizing model outputs.

The Reliability Frontier

This reordering of priorities is reflected by a stubborn reality on the ground. Billions of dollars have flowed into enterprise AI. Every major company has announced a strategy. The demos remain impressive, the benchmarks keep climbing, and actual productivity gains remain elusive. Recent studies tell the same uncomfortable truth: most organizations struggle to move beyond pilot projects. McKinsey reports that while 88% of enterprises are using AI, only 6-7% have achieved value at scale.[2] BCG's survey of 1,800 CxOs found that just 25% see significant AI value.[3] Gartner predicts 30% of generative AI projects will be abandoned after proof-of-concept by end of 2025.[4]

The paradox sharpens when set against the Poetiq result. Harness sophistication clearly works. Verification and refinement loops clearly extract more value from fixed models. The approach delivered a 20% accuracy improvement over the previous state of the art while cutting costs by more than half. If this is what harnesses can do, why does enterprise deployment keep stalling?

The answer lies in what made Poetiq's success possible. ARC-AGI-2 puzzles have a property that most enterprise work lacks: unambiguous verification criteria. A candidate solution either produces the correct output grid or it does not. The system can test its work against ground truth, detect failure with certainty, and iterate with clear signal. Most enterprise tasks offer no such clarity, and without it, the entire harness architecture collapses .

Chapter 2: The Accuracy Ceiling

Why Benchmark Performance Fails to Predict Deployment Success

Capability keeps climbing while deployment keeps stalling. Every few months, a new model surpasses its predecessors on established benchmarks, solving graduate-level science questions that stumped systems two years ago, conquering mathematical olympiad problems, completing coding challenges that once required human expertise. The trajectory is unmistakable and the progress is real. Yet organizations report impressive pilots and struggle to move AI into production. Scaling still works. Benchmark performance just measures the wrong thing.

Consider SimpleBench, a benchmark designed to test reasoning about everyday physical and social situations. The questions require no specialized knowledge: a juggler throws two balls and climbs a ladder; where are the balls now? Someone tapes a sandwich to a walking stick and moves to another room; how many whole sandwiches remain? These scenarios demand exactly the kind of common-sense reasoning that deployment requires. Unspecialized humans score 83.7%, while the best frontier models, as of early 2026, reach only the mid-70s.[7] The gap continues to narrow with each model generation, but the pattern it reveals matters more than the numbers themselves. Performance on tasks requiring memorized knowledge and pattern retrieval scales reliably with model size and training compute. Performance on tasks requiring robust world models and causal reasoning improves more slowly, through mechanisms that remain poorly understood.

Here lies the gap between benchmark success and deployment readiness. A system handling routine cases brilliantly while failing systematically on edge cases creates more liability than value, because the failures occur precisely when human judgment would be most needed and is least available.

Why Models Fail on Novel Problems

A comprehensive 2025 review by Zhang et al. examined how large language models contribute to scientific discovery, revealing a consistent pattern. LLMs have enabled genuine advances in drug combination discovery, protein structure prediction, and materials science, yet these systems remain limited as independent scientific

reasoners.[6] A gap persists between technical tool and creative engine, and bridging it requires infrastructure beyond the model itself.

The pattern across successful applications tells the same story: breakthroughs required sophisticated orchestration rather than raw model inference. LLM agents managed experimental workflows while retrieval pipelines provided access to current literature. Multi-agent debate systems refined hypotheses through structured disagreement, and automated validation loops provided ground-truth feedback. The models contributed generation capability; the infrastructure provided everything else, including direction, verification, error correction, and grounding in external reality. When researchers attempted to use models without this scaffolding, performance dropped sharply.

The underlying issue is statistical in nature. Models learn patterns from training data and recognize similar patterns at inference time, which works well when test cases resemble training cases and fails when they do not. Every benchmark eventually becomes training data, partly through direct contamination but more fundamentally because benchmarks sample from a distribution and models learn to recognize that distribution. Performance on MMLU improves because models encounter MMLU-like questions during training, develop representations that capture the structure of such questions, and retrieve appropriate reasoning patterns at inference time. This is exactly how statistical learning works, but it means that benchmark performance measures in-distribution accuracy while deployment requires out-of-distribution robustness.

The frontier of failure moves but persists. When models master one class of problems, researchers design harder benchmarks that probe capabilities the models lack, and performance drops. Then training adapts, performance rises, and the cycle repeats. This dynamic creates the illusion of steady progress toward general capability when it actually reveals a pattern of expanding competence within an ever-growing training distribution while genuine novelty remains challenging. A model that excels at olympiad problems encountered during training may still stumble on a problem requiring the same mathematical principles applied in an unfamiliar configuration.

Current progress follows a predictable path: larger models trained on larger datasets, expanding the distribution of problems they can handle. This approach has no theoretical ceiling, and with sufficient scale, models might eventually encounter training examples similar to almost any problem they face. But even if we could somehow train on everything, a deeper limitation remains. These systems are fundamentally stochastic, generating outputs by sampling from probability distributions. A stochastic process cannot guarantee a specific outcome with certainty; it can approach arbitrarily high reliability on familiar problems, but it cannot promise perfection, and it cannot reliably distinguish the familiar from the genuinely novel.

Why Reliability Degrades Over Steps

The stochastic nature of language models creates a second barrier that compounds the first, because single-step accuracy and process reliability are different quantities with different scaling properties. A model achieving 95% accuracy on individual decisions sounds impressive until you consider what happens over a sequence: twenty decisions at 95% accuracy each yield only 36% probability that all twenty are correct, fifty decisions yield 7.7%, and a hundred decisions yield 0.6%. Future scaling will not resolve this limitation.

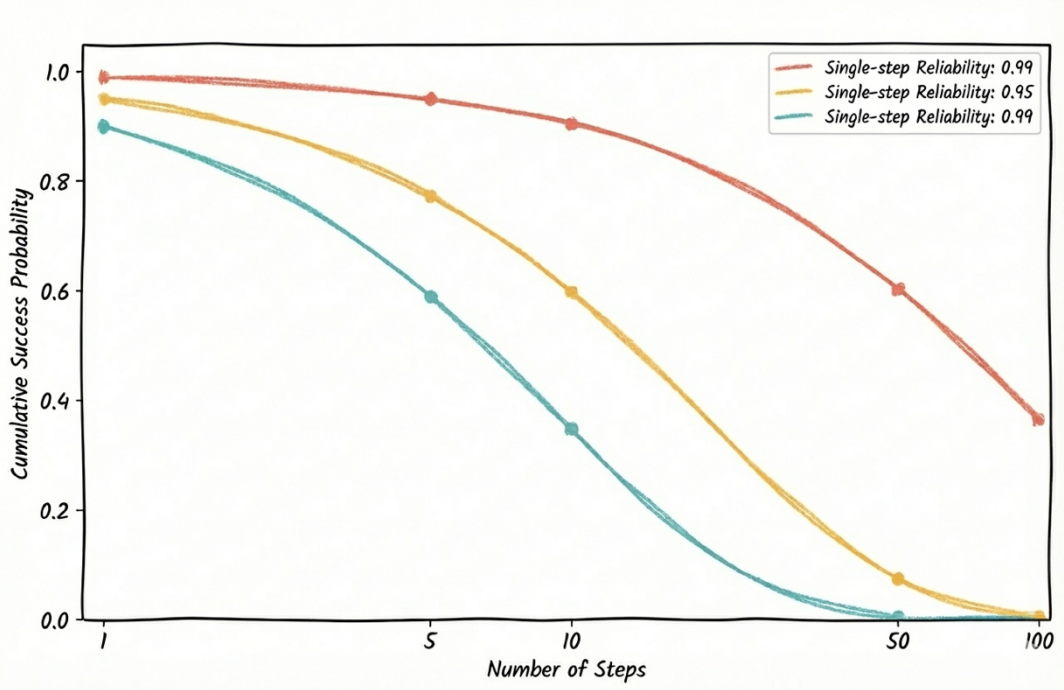

Figure 2: Degredation of Multi-Step LLM Reasoning Reliability

Consider how this plays out in a concrete workflow. An AI agent processing an expense report must extract the vendor name from a receipt, match it against the approved vendor list, verify the amount falls within policy limits, select the correct expense category, route to the appropriate approver based on department and amount, and schedule payment according to net terms. At 95% accuracy per step across these six decision points, roughly 73% of reports process correctly. Add the complexity that real enterprise processes demand, including currency conversion for international expenses, project code allocation, tax treatment determination, and duplicate detection, and the success rate falls to 60% across ten steps. The arithmetic is unforgiving, and real enterprise workflows routinely involve dozens of sequential decisions where a single error invalidates everything downstream.

Demis Hassabis has likened this behaviour to compound interest.[8] Agents become lost in accumulated context, unable to detect when they have drifted from objectives; they fall into loops, misinterpret termination signals, and lose track of their own state. These failures cluster at points where the system must integrate information across long spans and maintain goal-directed behavior without external anchoring.

The compound probability problem becomes severe precisely as we ask AI to do more ambitious things. A simple query-response interaction involves few decision points and tolerates occasional errors, while an autonomous agent executing a multi-step business process involves dozens or hundreds of sequential choices where a single error can cascade through every subsequent step. The same model that performs admirably in conversation may fail catastrophically when given extended autonomy; per-step capability remains unchanged, but process reliability demands a higher standard than single-step accuracy can provide.

The Self-Verification Paradox

Both barriers would be manageable if models could detect and correct their own failures. A system that notices when it has generalized incorrectly could flag uncertain outputs for review; a system that detects coherence drift could checkpoint its state and recover. But models cannot reliably perform either function,[20] and better training will not fix this because the limitation is structural, rooted in how these systems work.

The standard approach to improving AI reliability involves having the system check its own work: generate a solution, critique it, refine it, repeat. This approach yields diminishing returns because of a fundamental limitation: the verifier is built from the same statistical patterns as the generator.

Consider the 2025 International Mathematical Olympiad. Huang and Yang[10] constructed a verification-and-refinement pipeline using a publicly available frontier model. Without the pipeline, using a best-of-32 selection strategy where the model generates multiple candidates and chooses the most promising, accuracy on IMO problems reached 31.6%. With the verification pipeline, where the model generates solutions, critiques them, identifies errors, and iteratively refines its work, accuracy rose to approximately 85%: five of six problems solved, gold medal performance. The improvement is substantial, nearly tripling baseline accuracy through self-critique alone.

But the sixth problem proved insurmountable. Problem 6, designed to require genuine insight rather than pattern application, defeated every approach the researchers tried. The verification pipeline could catch errors on problems within the model's distribution, but it could not help when the problem itself lay outside what the model had learned to recognize. The same representations that failed to generate a correct solution also failed to verify one.

Call this the self-verification paradox. Homogeneous verification, where the same model generates and verifies, can raise the floor of performance by extracting more of what the model already knows, but it cannot raise the ceiling. The model's blind spots persist through any number of self-critique iterations. If the model learned to conflate correlation with causation, it will make that error when generating and fail to catch it when verifying, because the error is invisible to the system that learned it as truth.

Domains without clear success criteria make this worse. In mathematics, a proof either holds or it does not, and this binary clarity is precisely what allowed the verification pipeline to improve performance so dramatically on five of six problems. In strategic planning, success depends on competitor responses, market shifts, and execution quality, all of which resist formal verification. When the verification system relies on the same pattern-matching capabilities as the generator, it inherits identical

blind spots and cannot distinguish confident competence from confident hallucination.

The barriers documented here define one dimension of the reliability frontier: the gap between benchmark accuracy and deployment-grade consistency. Solving them would require architectural innovations beyond model scale, mechanisms for grounding outputs in external truth rather than internal pattern-matching.

Chapter 3: The Preference Gap

Why Formal Reliability Is Not Enough

Even if every formal capability gap were closed, enterprise work would face a second reliability dimension that verification cannot address.

Two equally competent lawyers draft different contracts because they have different judgment about what matters. Two skilled marketers propose different campaigns because they have learned different lessons from experience. The value these professionals provide comes not just from their knowledge but from their accumulated sense of what works in their specific context. A contract that is formally correct may still be wrong for this client, this deal, this lawyer's sense of appropriate risk allocation. A marketing campaign that follows best practices may still miss the distinctive voice that makes this brand recognizable.

Current AI systems cannot access these preferences. They generate outputs from general patterns learned during training, patterns that reflect average tendencies across millions of documents rather than specific judgment of specific users. The result is work that is competent in a generic sense but fails to satisfy in a specific sense. Prompt engineering offers limited help. You can describe preferences in a system prompt, include examples of past work, specify requirements in exhaustive detail. The output remains generic because the model does not know you. Each interaction starts fresh, with no memory of what worked before, no accumulation of feedback that would allow preferences to emerge from use rather than explicit instruction.

This is the preference gap. Closing it requires something fundamentally different from the architectural innovations that might address the accuracy ceiling.

The Three Loops Framework

Understanding how formal reliability and preference reliability map onto actual work clarifies where each matters and where the frontier lies.

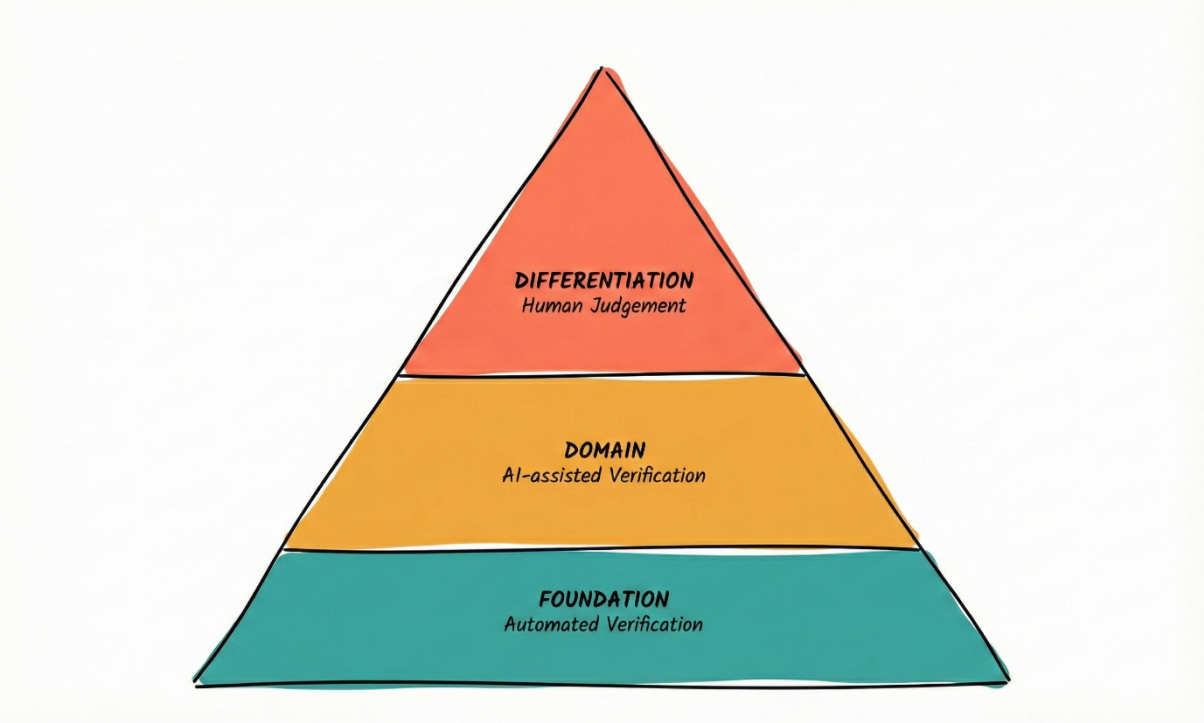

Figure 3: Three Loops of Work Verification

At the base sits Foundation: work where correctness is objectively verifiable. Code that compiles or does not. Compliance checks against codified rules. Data transformations that validate against schemas. For these tasks, formal reliability is the binding constraint. If the system can produce accurate outputs consistently, it can verify its own work against external criteria and iterate until the criteria are met. The feedback loop closes without human judgment. Progress on the accuracy ceiling unlocks immediate value here.

The middle layer is Domain: work requiring specialized expertise but following assessable structure. Legal analysis that must cite precedent correctly. Medical synthesis that must align with clinical evidence. Technical documentation that must be internally consistent. Formal reliability remains necessary but insufficient. The system must get the facts right, but it must also exercise judgment about which facts matter, how to frame them, what implications to draw. A pharmaceutical company preparing a regulatory submission can use AI systems that check whether cited sources support claims made, look for logical gaps, verify formatting requirements. When problems surface, the system iterates before any human sees the output. Yet the final submission still requires human judgment about emphasis, framing,

strategic positioning. Preference reliability begins to matter alongside formal reliability, though the structured nature of the domain provides partial anchoring.

At the apex sits Differentiation: work where quality depends on taste, strategy, and context that cannot be codified. Brand voice. Strategic positioning. Creative direction. No external criteria can verify these outputs. Only human judgment suffices, and human judgment varies across practitioners who may legitimately disagree. A consumer products company developing a marketing campaign cannot specify rules that reliably produce the exceptional concept. The exceptional concept feels right to the people who understand this brand, this market, this moment. Experienced marketers recognize this quality when they see it, but they cannot specify rules that reliably produce it. That recognition lives in accumulated preferences that no prompt can capture.

The two reliability dimensions bind differently across these layers. Formal reliability constrains all three but most tightly at the base, where verification can close the loop. Preference reliability constrains the upper layers, where formal verification cannot help because there is no objective standard to verify against. The accuracy ceiling matters most for Foundation work. The preference gap matters most for Differentiation work. And the gap between current AI and genuine enterprise value lives primarily in that third layer, where the work that commands premium fees and creates competitive advantage resides.

The Preference Learning Challenge

The deeper challenge involves how organizational work actually happens. A company's output emerges from many individuals, each contributing their own judgment and taste. A single company-wide AI approach fails because it tries to average across people who legitimately disagree about quality. The senior designer and the junior copywriter have different instincts, and both contribute to the final product. The general counsel and the business development lead have different risk tolerances, and the company needs both perspectives.

This creates the preference learning challenge. Generic AI fails because it cannot capture individual judgment. But individual judgment varies legitimately across practitioners, which means there is no single correct preference to learn. The system must somehow learn what specific individuals consider good without sacrificing the

general knowledge encoded in foundation models, and it must do so for many individuals whose preferences may conflict.

Current systems meet none of these requirements. They lose track of stated preferences within a single conversation, let alone across sessions and tasks. The judgment that would make AI genuinely useful never accumulates. No learning persists. Closing this gap requires architectures that treat preference learning as a first-class concern rather than an afterthought.

Part II: What Scales Next

Chapter 4: The Harness Paradigm

From Model-Centric to System-Centric AI

The preceding chapters established a diagnosis. Foundation models have achieved extraordinary fluency, yet that fluency has not translated into the reliability that enterprise deployment requires. The core issue is not that models plateau at some fixed accuracy ceiling. They will likely break through current benchmarks as scaling continues. The issue is that single-step accuracy and process reliability are different quantities with different scaling properties. A model achieving 95% accuracy on individual decisions yields only 36% probability of success across twenty sequential steps. Even 99% per-step accuracy, a level no current system approaches on complex tasks, compounds to just 37% success over a hundred decisions. The math does not care about model size. It cares about error rates and step counts. Enterprise workflows routinely involve dozens of sequential decisions where a single error cascades through everything downstream. This is the reliability problem that model scaling alone cannot solve.

This chapter proposes the architecture that addresses that limitation. As Chapter 1 established, AI capability scales along two axes: Model Scale and Harness Sophistication. The industry has invested hundreds of billions in the first while leaving the second largely unexplored. Poetiq's ARC-AGI-2 result demonstrated what becomes possible when that imbalance corrects. The foundation model was the engine. The harness was the driver.

The Frozen Foundation Principle

Central to this architecture is a counterintuitive constraint: the foundation model's weights remain unchanged during operation. We call this the Frozen Foundation Principle. Models become interchangeable substrates, not fine-tuning targets.

The constraint is strategic, not ideological. Fine-tuning creates a well-documented failure mode. When a model learns new information through weight updates, it degrades performance on previously mastered tasks. Research has documented the severity of catastrophic forgetting during fine-tuning: standard fine-tuning can erode 20-40% of a model's accuracy on specific prior benchmarks, while safety alignment can degrade by up to 90% [11,12] Even sophisticated mitigation techniques reduce this

degradation to roughly 11%, a rate that compounds disastrously over time. An organization that fine-tunes weekly for a year accumulates massive erosion of general capability. The model becomes expert in narrow patterns while losing the broad reasoning that made it useful.

Freezing the foundation model eliminates this failure mode entirely. No weights update means no forgetting. The system's adaptation to organizational context happens through mechanisms that sit outside the model. These include explicit preferences stored in lightweight adapters, verification rules encoded in separate systems, and iteration strategies learned through meta-optimization. Such mechanisms can be inspected, versioned, and rolled back. A fine-tuned model mutates in ways that are difficult to trace.[13] A frozen model wrapped in explicit harness logic produces behavior that can be audited.

This creates a clean separation of concerns. Model providers optimize general capability, which includes the raw creative spark, the breadth of world knowledge, and the fluency of generation. Harness builders optimize orchestration, verification, and adaptation, which is the machinery that transforms raw capability into reliable outputs. Investment flows to whichever layer currently offers higher returns. When model capability is the bottleneck, model investment dominates. When verification is the bottleneck, harness investment dominates. The evidence suggests we have entered the second regime.

Propose and Verify

Intelligence in this paradigm emerges from the interaction between the generator and the harness. The foundation model provides raw capability, including the creative spark, the linguistic fluency, and the broad retrieval of facts accumulated through pre-training. The harness provides structure, including the critical eye, the stopping condition, the memory of preference, and the contact with external ground truth.

Neither component suffices alone. A powerful model without verification produces confident errors at scale. A sophisticated harness without a capable model has nothing to shape. The model’s job is to propose. The harness’s job is to decide when a proposal meets the bar.

This understanding reframes the path to reliable enterprise AI. Organizations do not need to wait for a perfect model that never hallucinates. That model is not on any credible roadmap. They need to build, or acquire, architecture that assumes the model will err and possesses the machinery to detect and correct those errors before they reach users or decisions. The verification gap is not a temporary limitation awaiting the next training run. It is a structural feature of how autoregressive language models work. Closing it requires architectural innovation, not just scale.

The harness paradigm does not reject model scaling. It reorders the bottleneck. Model scale improves what the generator can produce. Harness sophistication determines what the system can trust. The past two years have made the gap between those curves visible. Fluency has outpaced verification. The result is technology that dazzles in demonstrations and stalls in deployment. A harness is the architecture that closes that gap.

Chapter 5: The Harness Architecture

How to Close the Reliability Loop

Chapter 4 established that a foundation model is an engine, not a complete vehicle. Raw generative capability has become cheap enough, and good enough, that the limiting factor in enterprise deployment is no longer whether a model can produce an answer that sounds plausible. The limiting factor is whether a system can take responsibility for that answer before it reaches a human decision point. That responsibility has concrete meaning. It means iterating when iteration is warranted, adapting outputs to the judgment of specific people without destabilizing the underlying substrate, and checking claims against something outside the model's own sense of plausibility.

A harness does not add one feature. It adds a control loop. The generator proposes. The harness evaluates. The harness learns. The generator proposes again under tighter constraints, with better guidance and clearer failure information. Over time, the system becomes less impressed by fluency and more constrained by reality.

This chapter details the three components that make that loop work in practice: the orchestrator, which governs iteration, strategy selection, and stopping; a preference learning system that captures judgment and taste without touching foundation weights; and a verification layer that spans two very different regimes. In closed domains, verification can be decisive. In complex domains, verification becomes probabilistic and simulation-based, with the goal shifting from certainty to calibrated confidence.

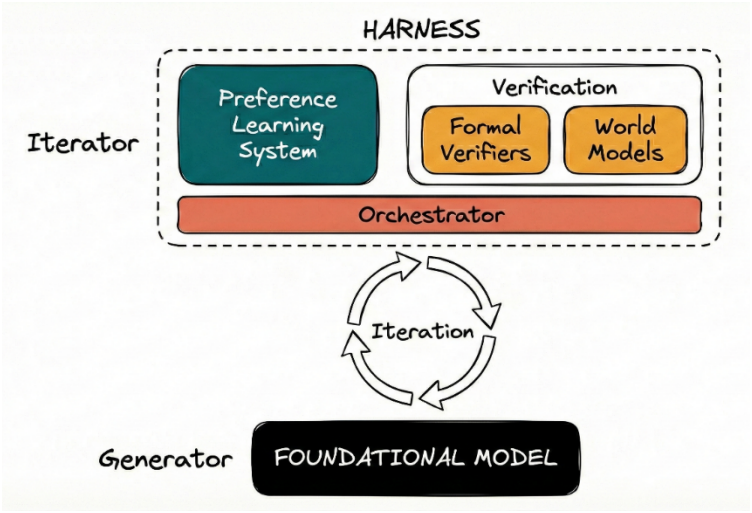

Figure 4: The Harness Architecture

The Orchestrator

At the center of the harness sits a control problem. A foundation model can generate candidates at high speed, but it has no intrinsic reason to stop generating. It has no native sense of which strategy is appropriate for a given task, and no reliable internal signal that distinguishes a correct answer from a persuasive one. The orchestrator supplies that missing control structure. It decides how many attempts to make, what kinds of attempts to make, which tools and checks to invoke, how to interpret failures, and when additional compute has stopped buying improvement.

In its simplest form, orchestration looks like a familiar loop. Generate a candidate, evaluate against available checks, extract failure information if the candidate fails, generate a revised candidate. A naïve implementation already improves outcomes because it forces the model to confront errors instead of defending its first instinct. The interesting systems go further. They treat the loop itself as an object of optimization.

Poetiq's ARC-AGI-2 result offers the clearest illustration. The performance gain did not come from brute-force sampling or from a single prompting trick. Their system behaved like a meta-solver. It carried a library of strategies and learned which ones were effective for which problem families. It used verification to decide whether it had succeeded, then stopped aggressively when the evidence was sufficient. On many tasks, it averaged fewer than two model invocations. That is the signature of an

intelligent controller. A system that is unsure keeps talking. A system that can recognize success stops.

For most of the last decade, the dominant economic lever in AI was training compute. The cost was paid upfront, amortized across users. Harness systems shift a meaningful portion of capability improvement into inference-time compute. The system spends extra thinking and extra verification on the cases that demand it, and it does so only until the checks converge. That change matters because it makes "thinking time" a tunable variable. When the output will be read by an executive committee, the system can run a longer loop. When the output is a low-stakes draft, it can stop early. Reliability becomes a budgeted choice rather than a fixed property of a model checkpoint.

Research on test-time compute makes the tradeoff concrete. Snell and colleagues at DeepMind demonstrated that scaling inference intelligently can match the performance of models fourteen times larger[14], provided the inference budget is allocated optimally rather than spent uniformly. The key insight is that compute-optimal strategies vary by problem difficulty. Easy problems benefit from sequential revision. Hard problems require parallel candidate generation followed by selective refinement. An intelligent orchestrator learns those regimes through experience.

The practical lesson for enterprise deployment is not that inference-time scaling is always efficient. The lesson is that a system which controls inference-time compute can buy reliability without waiting for the next training run, and it can do so in a way that is legible. Token usage, tool calls, and verification steps can be measured, priced, and allocated by policy. A legal team can define a "draft" mode that runs fast checks and a "submission" mode that runs deep checks. A product team can define a "prototype" mode that accepts higher variance and a "release" mode that demands convergence. Without an orchestrator, those policies remain slogans. With one, they become enforceable.

Strategy Selection and Adaptation

A harness that uses the same approach for every task quickly becomes brittle. Orchestration requires a portfolio of methods: different prompting styles, different decomposition patterns, different model routing policies, different verification intensities. Each has regimes where it wins. A strong orchestrator learns those

regimes through experience. It records outcomes, failure modes, costs, and time-to-convergence. Over time, it becomes less like a general chat interface and more like a practiced operator that knows which moves are wasteful.

Strategy adaptation operates at two levels. First, the orchestrator learns which prompting strategies work for which problem classes. Code generation benefits from test-first approaches where the model generates unit tests before implementation. Legal analysis benefits from issue-spotting prompts that surface potential weaknesses before drafting. Strategic planning benefits from red-teaming simulations where the model argues against its own proposals. The orchestrator does not apply a single method universally. It matches method to problem type.

The second level involves sub-model selection. Different foundation models have different strengths. One model may excel at creative ideation but occasionally hallucinate technical details. Another produces cautious, well-cited analysis but struggles with speculative reasoning. A third balances the two but carries higher per-token costs. The orchestrator routes queries to the appropriate substrate, sometimes invoking multiple models in parallel and synthesizing their outputs. This model agnosticism is central to the frozen foundation principle. When new models appear, the orchestrator incorporates them without retraining, shifting inference patterns as capabilities evolve.

AlphaCodium, developed by CodiumAI, provides a useful supporting case because it shows how orchestration exploits asymmetry. In many programming problems, generating correct code is hard, but generating tests that expose incorrect code is easier. AlphaCodium turned that observation into a flow. It asked the model to write candidate solutions, then asked the model to generate additional tests that would likely fail those solutions, then used the failures to drive refinement. The result was a dramatic gain in accuracy. GPT-4's score on CodeContests improved from 19% to 44% using roughly fifteen to twenty model calls[15], a tiny fraction of the brute-force generation budgets that earlier systems used. The improvement came from control, not from scale. The orchestrator forced the generator to live inside a tightening cage of evidence.

Stopping Criteria

Stopping is the most important decision the orchestrator makes, and it is the easiest to get wrong. Systems that stop too early ship subtle errors. Systems that stop too late waste compute and still ship errors because the loop degenerates into variation without progress.

A reliable orchestrator ties stopping to evidence. In closed domains, that evidence can be decisive. The code passes a test suite that the model did not write and cannot influence. The schema validates. The theorem checker accepts the proof. In open domains, the evidence is weaker, which makes stopping harder but not optional. The system must decide when uncertainty is irreducible under the available information and budget, then surface that uncertainty explicitly rather than hiding behind extra prose.

Poetiq's self-auditing module illustrates the principle. It monitored progress during the iteration loop and halted generation once a solution met verification criteria, averaging fewer than two requests per problem.[16] This intelligence prevents the cost explosion typically associated with agentic workflows. The system does not just loop. It converges.

Orchestration Today

Here is the deeper pattern. Orchestration converts a model's strengths into constraints on its weaknesses. Language models are strong at generating variants, articulating rationales, enumerating edge cases, and proposing tests. They are weaker at truth-tracking in a single pass. An orchestrator turns those strengths into a process that is difficult to game by accident. The system does not ask the model to be right once. It asks the model to survive a sequence of checks that grow more demanding as the budget increases.

The market has recognized orchestration's importance. Agentic coding tools like Claude Code and oh-my-opencode already implement sophisticated versions of the patterns described above. They maintain context across multi-step tasks, invoke external tools based on intermediate results, adapt their approach when initial attempts fail, and manage complex workflows that would have seemed like research prototypes two years ago. These systems ship to developers today. They work. The question is no longer whether orchestration matters but how far it can carry reliability on its own.

Of the three harness components, orchestration has received the most attention and the most capital. The techniques are understood. The engineering patterns are established. Preference learning and verification remain underbuilt relative to their importance. Most deployed systems still treat preference as a prompt engineering problem and verification as an afterthought. An orchestrator that generates twenty candidates means nothing if the system cannot distinguish which one this user would choose, or whether any of them are true.

The Preference Learning System

Orchestration can close the reliability loop for tasks with objective checks, but it does not solve the enterprise gap on its own. Enterprises rarely fail to adopt AI because they cannot generate text. They fail because the text is generic, and because the system cannot learn what "good" means for the people doing the work.

A marketing director does not want the average marketing copy. A senior counsel does not want the average contract clause. In high-value work, competence is table stakes. Differentiation lives in judgment. That judgment is difficult to capture because it is rarely expressed as rules. Experts often cannot explain why they prefer one draft over another beyond a few surface rationales. Their real criteria are compressed into years of experience and tacit pattern recognition. Asking them to write a specification for their taste yields a brittle proxy that fails on the cases that matter.

A preference learning system—nascent in current implementations but directionally clear in research—avoids that trap by learning from behavior, not declarations.

Selection-Based Training

The simplest signal is choice. Present multiple candidates that satisfy the same objective constraints and ask the user to pick. The user's selection becomes training data. Over many selections, the system learns a model of preference that is specific to that person, that role, and that organization. The preference model does not need to be perfect. It needs to be directionally correct often enough that the system's first draft steadily improves, and that the system's refinements converge toward what the user would have written.

Preference learning is most effective when it is embedded into normal work rather than treated as an extra annotation burden. A lawyer already compares alternatives. A strategist already rejects two drafts before endorsing the third. A product manager already edits. Those actions are signals. The harness can treat them as data if it has a way to log them, normalize them, and learn from them.

Direct Preference Optimization and related methods provide a clean way to train on these signals without building a separate reward model. The training objective is straightforward. Increase the probability of outputs a user selects and decrease the probability of outputs the user rejects, conditioned on the same prompt and context. The implementation detail matters less than the behavior it enables. The system must get better through use, and it must do so without eroding its general capabilities.

The memory features now common in consumer AI chatbots offer a useful contrast. These systems remember facts about users across conversations: names, preferences stated in passing, projects mentioned weeks ago. The capability sounds like preference learning. It is not. Memory systems accumulate declarative facts. A preference adapter learns latent structure.

The difference becomes obvious under pressure. A memory system might record that a user "prefers concise writing." But concise means different things in different contexts. A legal memo requires different concision than a marketing headline. The memory system applies the preference uniformly because it stores a label, not a model. It tries to generalize where specificity is required. A proper preference adapter learns that this user wants aggressive compression in status updates, moderate density in analysis documents, and expansive detail in technical specifications. It learns the boundaries between those contexts from observed choices, not from a single declared rule.

Memory systems also suffer from scope creep. They accumulate everything that might be relevant without learning what actually predicts preference. The result is a growing pile of context that the model must somehow weigh, with no principled way to determine which facts matter for which decisions. A preference adapter, by contrast, learns weights. It discovers that for this user, tone matters more than structure in client communications, while structure dominates in internal

documents. The adapter does not just remember that tone matters. It learns how much tone matters relative to other factors, in which contexts, for which output types.

The distinction is architectural. Memory augments the prompt with retrieved facts. Preference learning shapes the probability distribution over outputs. One tells the model what you said you want. The other teaches the model what you actually choose.

Accumulating Expertise

Preference learning also changes how organizational knowledge accumulates. In most enterprises, judgment lives in people. When they leave, a portion of that judgment leaves with them. Written process documents capture a thin slice.

A preference system can capture more. Over time, it builds a record of what the organization chose when confronted with real alternatives under real constraints. That record is more valuable than a style guide because it encodes tradeoffs rather than slogans.

Teams often try to create a single "company voice" adapter and then wonder why the system feels bland. Organizations are not monoliths. A senior partner's preferences differ from a junior associate's. A design lead's taste differs from a brand manager's. A product organization often benefits from preserving those differences because they reflect legitimate roles and incentives.

Preference learning should therefore be personal first, then compositional. The harness can maintain multiple preference models and apply them based on workflow. A draft can be generated under one adapter, reviewed under another, and finalized under a third. The artifact that emerges is not an average. It is a negotiated output shaped by distinct judgments.

Governance matters here because preference data is sensitive. An enterprise preference system must make clear what is being learned, where it is stored, and how it can be erased or reset. It must support versioning because preferences drift. A leader's priorities change with market conditions. A legal team's risk tolerance changes after a regulatory event. The preference layer must adapt without rewriting history, and it must explain what it believes the user tends to prefer. If preference

learning becomes a black box, organizations will treat it as yet another un-auditable model. If it remains explicit and controllable, it becomes infrastructure.

Compounding Effects

The interaction between preference learning and orchestration is where the compounding effect appears. Orchestration can generate many candidates cheaply. Preference learning can rank those candidates with increasing accuracy. The system begins to spend its verification budget on the candidates that the user is most likely to accept, which reduces waste and accelerates convergence.

A virtuous cycle emerges. As the preference adapter learns an individual's judgment, the orchestrator generates candidates in a narrower, more relevant space. Fewer candidates are discarded. The loop tightens. What once required five iterations now requires two. What once felt like wrestling with a generic system starts to feel like collaborating with an assistant that understands the difference between acceptable and excellent for this specific person.

The compounding extends across time. Early interactions with a preference adapter may feel only marginally better than baseline, because the adapter has not yet accumulated enough signal. But as choices accumulate into hundreds, then thousands, the model of preference becomes increasingly precise. The harness begins to anticipate not just what the user prefers but what they would prefer in situations they have not yet encountered, generalizing from patterns in their past choices.

Here is where the enterprise gap begins to close. Generic AI produces work that is competent in a generic sense but fails to satisfy in a specific sense. A harness with orchestration and preference learning produces work that is competent in a specific sense, reflecting the accumulated judgment of the people who will use it. The system does not replace expertise. It encodes and amplifies it.

But taste without truth produces pleasing fiction. A system that learns what you prefer but cannot verify whether what you prefer is true remains dangerous. Preference learning addresses what orchestration alone cannot capture. Verification addresses what neither can provide alone.

The Verification Layer

The preference system solves "what do you want." Verification solves a different question: "is it true." In enterprise deployment, the second question is the gate that determines whether autonomy is possible at all.

Verification begins with a constraint that Chapter 2 made explicit. Homogeneous self-critique cannot be the foundation of trust. A model can critique what looks wrong to the same statistical machinery that produced the output. It struggles with errors that look plausible. A verification layer must therefore be anchored outside the generator's sense of plausibility.

Understanding verification as a spectrum rather than a single technique is essential. The difficulty of verification scales with how much of the world the verifier must model to do its job. At one end, formal verification operates in closed systems where the rules are complete and the specification is finite. At the other end, verification of open-ended claims about strategy or policy would require modeling the world itself, understanding causal mechanisms well enough to predict consequences under novel interventions. Current systems occupy positions along this spectrum, with reliability varying accordingly. The sections that follow examine two positions on this spectrum where deployed systems can operate today: formal verification in closed domains and parameterized simulation in complex ones.

Formal Verification for Closed Domains

Closed domains are those where outputs can be evaluated against explicit specifications. Software is the archetype. Code compiles or it does not. Unit tests pass or they fail. Static analyzers flag violations or they remain silent. Data pipelines can be validated against schemas. Compliance checks can be expressed as rule sets. Mathematics, when translated into a formal language, can be verified against a proof kernel.

In these domains, verification is an external constraint that the generator cannot talk its way around. A compiler does not accept persuasive explanations. A theorem prover does not accept elegant prose. The verifier enforces a standard that is orthogonal to the generator's fluency.

DeepMind's AlphaProof provides the paradigm case. The system combines a Gemini-fine-tuned language model with AlphaZero reinforcement learning, using the Lean

proof assistant as its verification backend. The neural component proposes proof tactics given the current proof state. The symbolic component, Lean's kernel, checks each step against mathematical axioms and inference rules.

This result stands in sharp contrast to the homogeneous verification discussed in Chapter 2. Recall that the 2025 IMO verification pipeline using self-critique achieved gold-medal performance on five of six problems but failed completely on Problem 6—the same problem that defeated nearly all human contestants. That failure illustrated the self-verification paradox: the model's blind spots persisted through any number of self-critique iterations because the verifier shared the generator's limitations. AlphaProof succeeds where homogeneous verification fails because the verification comes from outside the generator's statistical patterns. Lean's kernel does not pattern-match; it checks proofs against axioms. The heterogeneous architecture breaks the paradox by grounding verification in formal logic rather than learned plausibility.

The system cannot hallucinate an incorrect proof because incorrect proofs do not compile. A missing lemma is not waved away. If the checker accepts the proof, the proof is valid by construction. At the 2024 International Mathematical Olympiad, AlphaProof earned silver-medal points by solving four of six problems[17], including Problem 6, which only five of 609 human contestants solved completely.[18] The neural generator provided intuition. The formal verifier provided certainty.

The engineering consequence is straightforward. In closed domains, the harness should force the generator to speak in a language the verifier can check. That often means translating vague intent into formal artifacts. A policy recommendation becomes a set of compliance assertions. A data transformation becomes code plus a schema contract. A mathematical argument becomes Lean. The orchestrator can use the failure messages from the verifier as a high-quality gradient for refinement. When the checker rejects the artifact, it often supplies a counterexample, a type mismatch, or an explicit gap. Those signals are far more useful than a model's self-critique because they are grounded in a system that does not share the model's blind spots.

Software verification admits a second important trick, one that AlphaCodium demonstrated: tests are a form of specification. They are imperfect, but they can be

generated and expanded. A harness can ask the model to write code, then ask the model to generate tests that would break that code, then iterate until the code survives. The success of this flow depends on the existence of a test suite that the generator cannot easily tailor to. Holding out tests, adding property-based testing, adding fuzzing, and mixing human-written tests with model-generated ones all help. A model can overfit to a fixed set of visible tests. It struggles when the harness keeps widening the net.

Formal verification does not make the system omnipotent. It makes it accountable within the domain where specifications are real. That distinction matters because it prevents overreach. A harness that tries to force every domain into formal proof becomes brittle. A harness that uses formal methods where they fit can push reliability toward the levels enterprise autonomy requires.

Parameterized Simulation for Complex Domains

Most of the decisions enterprises care about most do not live in closed domains. A market entry strategy cannot be compiled. A policy memo has no theorem checker. These tasks resist formal specification because their correctness depends on a world that changes, reacts, and contains unknowns.

Verification in complex domains therefore changes shape. The goal shifts from proving correctness to stress-testing under structured uncertainty. Instead of a binary verdict, the harness produces outcome distributions across scenarios, with explicit assumptions and quantified sensitivity. It tries to answer the question executives actually need answered: under what conditions does this recommendation fail, and how badly?

Parameterized simulation, which is a proto world model of sorts, is the most reliable path for this kind of verification in the current state of the art. The structure of the model is specified by domain expertise. The parameters are calibrated to empirical data. The assumptions are explicit, stated in documentation rather than implicit in neural weights.

For a market entry decision, the structure might include adoption curves by channel, competitor response patterns, pricing elasticity, logistics constraints, and tariff scenario trees. The parameters come from historical datasets, internal sales data,

public trade statistics, and expert estimates with documented priors. The harness then runs Monte Carlo simulations, agent-based simulations, or systems dynamics models to produce outcome distributions rather than point forecasts.

When parameterized simulation fails, it fails in an interpretable way. The assumptions are explicit. The parameters can be inspected. Sensitivity analysis can reveal whether the recommendation rests on one fragile number. In enterprise settings, that interpretability matters. The harness is not asked to predict the future with certainty. It is asked to make uncertainty legible enough that humans can decide where to place bets and where to gather more information.

A practical parameterized simulation therefore looks like a disciplined financial analyst. It forces the generator to state assumptions clearly. It builds a causal sketch of the domain, even if the sketch is incomplete. It separates what is supported by evidence from what is an inference. It runs counterfactual scenarios. It reports distributions and sensitivities. It treats unknowns as first-class objects rather than burying them inside confident prose.

This approach transforms strategic recommendations from persuasive narratives into testable hypotheses. It surfaces the assumptions that would need to be true for a strategy to succeed. It identifies sensitivities, the variables that dominate variance in outcomes. It stress-tests against historical patterns of competitive response, market shifts, and execution risk. It provides decision support under structured uncertainty rather than false confidence.

The limitations are equally important to understand. The models are built from historical patterns and expert-encoded structural assumptions. They can explore variations on what has happened. They struggle with genuinely novel scenarios: competitive moves that have no historical precedent, market structures that emerge from technological discontinuity, interventions that change the system being modeled. The structural assumptions themselves remain unvalidated by the simulation process. The simulation checks recommendations against a model of the world. It does not check recommendations against the full complexity of the world itself, but a fuzzy representation capture through the minds of humans.

Parameterized simulation outputs calibrated confidence under stated assumptions, not binary verdicts. A strategic recommendation returns not "approved" but a

probability distribution of outcomes conditional on the encoded assumptions about competitor response timing and tariff stability. These probabilistic assessments are more useful than false certainty because they encode model limitations explicitly. They force human decision-makers to confront uncertainty rather than hiding it behind authoritative prose.

Enterprises do not need a single answer with a false aura of certainty. They need a decision-support package that makes risk legible and assumptions explicit. The harness provides that package. It does not provide guarantees about open-world outcomes that no current system can deliver. Chapter 6 examines why that gap exists, and what closing it would require.

Worked Example: Market Entry Under Uncertainty

To see how the three components interact, consider a concrete decision that sits in the middle of enterprise complexity. A European footwear manufacturer is evaluating expansion into South America. The company has strong product-market fit in several European markets and a resilient supply chain, but limited brand recognition outside its home region. The decision carries real risk. Currency volatility can erase margin. Tariff regimes can shift with elections. Competitors can retaliate with discounting or channel capture.

A naive AI system will produce a strategy memo that sounds professional and still be dangerous, because the danger lives in the tail scenarios and the second-order effects. The harness transforms this into a verified decision-support process.

Component 1: Foundational Model Generation

The harness begins where every system begins, with generation. The foundation model ingests the company's constraints and goals. It receives unit economics, production capacity, current distribution channels, brand positioning, and a time horizon. It is asked to propose multiple strategies, not one.

Four candidates emerge. A direct export plan through regional distributors. A local distribution partnership with shared economics. An acquisition of a regional brand with existing channels. An e-commerce-first entry with limited physical footprint. Each strategy includes an execution sequence, resource requirements, expected time to traction, and an initial risk register.

A single-pass chatbot would stop here. The harness does not. The orchestrator treats these candidates as hypotheses. It establishes a budget and a plan for evaluation. It decides what must be true for each strategy to work, then assigns verification work accordingly. It requires that every strategy expose its critical assumptions, that key claims be grounded in cited data, and that forecasts be expressed as distributions rather than point estimates. It generates additional variants where the initial candidates are too similar, because diversity in strategy space is valuable before narrowing begins.

Component 2: Preference Adapter

The company's strategy leader has worked with an embedded preference module over several months. Through iterative feedback on previous strategic recommendations, thousands of small choices about which analyses to develop, which risk framings to accept, which presentation styles to adopt, the adapter has learned this executive's priorities.

This particular executive consistently prefers capital-light approaches over fixed investments. She tolerates moderate brand dilution when speed matters. She weights supply chain resilience above near-term growth. She has low patience for integration complexity and high sensitivity to currency exposure. These preferences are not a paragraph in a prompt. They are an accumulated pattern of choices.

The harness uses the preference adapter to score the candidate strategies against that pattern. The acquisition strategy, while potentially high-upside, triggers the executive's demonstrated aversion to integration risk. The direct export strategy aligns with capital discipline but conflicts with her low tolerance for logistics fragility in unfamiliar markets. The e-commerce-first strategy aligns with speed and capital-light constraints but raises concerns about returns, sizing, and last-mile reliability that past feedback suggests she finds unacceptable without mitigation. The partnership strategy looks most promising, provided partner incentives can be aligned.

The adapter does not only rank. It refines. The partnership strategy is rewritten to reduce capital exposure further. The e-commerce-first strategy is rewritten to include a staged rollout and a plan for handling returns locally. These are still drafts, but they are drafts that reflect accumulated judgment rather than generic strategy templates.

Component 3: Parameterized Simulation

The harness now asks a different question. Given these refined strategies, what happens when we stress-test them against structured uncertainty?

The simulation begins with explicit structure. The harness uses a hardcoded model that reflects the company's context as understood from historical patterns and available economic models of the companies experts. Demand depends on channel reach, brand awareness, price positioning, and local consumer preferences. Margin depends on production cost, shipping, tariffs, returns, and currency. Competitor response affects pricing, channel access, and marketing efficiency. Execution risk affects time-to-traction and burn.

The harness ties this model to data. It pulls historical currency volatility distributions for the relevant countries: ten years of data on the Brazilian real and Argentine peso, with explicit fat-tail modeling. It encodes tariff regimes and plausible policy changes as scenario trees calibrated to expert judgment on trade policy trajectories. It uses internal company data for return rates, unit margins, and production constraints. It uses market data for category growth, competitor share, and channel structures. Where data is missing, it forces priors into the open and marks them as such.

The orchestrator then runs simulations. Ten thousand scenarios are sampled across currency paths, tariff changes, demand trajectories, and competitor responses. Competitor response is modeled as a set of behavioral rules calibrated to past market entries in comparable categories. Some competitors defend share through price. Others defend through distribution exclusivity. Others ignore new entrants until traction is visible. The harness does not claim to know which competitor will act which way. It explores a distribution of responses plausible enough to expose the strategies that collapse under predictable pressure.

Output: Decision-Support Package

The output is not a single recommendation. It is a profile for each candidate strategy.

The partnership strategy shows the highest median outcome because it accelerates distribution without heavy capital investment. But it has a heavy left tail. In twenty-three percent of scenarios, outcomes turn negative, with failure clustering around

partner reliability issues and currency crises. The simulation identifies the critical assumption: partner quality matters more than market growth rates.

The e-commerce-first strategy has lower downside risk, only eleven percent of scenarios turn negative, but a slower path to meaningful scale unless marketing efficiency stays high. It is most sensitive to customer acquisition costs and return rates.

The export strategy is highly sensitive to currency paths and logistics costs. Its probability of success drops sharply under a plausible cluster of adverse scenarios involving simultaneous currency depreciation and tariff increases.

The acquisition strategy has the highest upside and the highest variance. Failure modes are dominated by integration complexity and channel conflict with the acquired brand's existing partners.

The harness reports outcome distributions, not point forecasts. It highlights sensitivities. It extracts the assumptions that dominate the variance. It suggests information-gathering moves that would reduce uncertainty where it matters. If the partnership strategy's left tail is driven by partner incentives, the harness proposes a pilot structure that tests partner performance quickly with clear termination rights. If the export strategy's fragility is driven by currency volatility, it proposes a hedging policy and shows how the distribution changes under different hedging costs. If competitor retaliation dominates, it proposes stress scenarios that simulate aggressive discounting and examines whether margin survives.

The executive receives strategies filtered through her own learned preferences, with counterfactual analysis showing how each performs across simulated futures. The decision remains human. The harness provides verified, preference-aligned options with quantified uncertainty.

Why the Three Components Are Inseparable

This worked example clarifies why the three components cannot be deployed in isolation. Without orchestration, the system would generate one memo and stop, never confronting the ways that memo might fail. Without preference learning, the system would produce generic strategy options that do not reflect the executive's actual tradeoffs, and it would waste verification budget exploring directions the

decision-maker will reject on sight. Without verification, the system would deliver confident prose that collapses under the first serious challenge, a strategy that sounds rigorous but has never been stress-tested.

The harness closes the loop by making generation provisional, by making taste learnable, and by making evaluation external. The system does not replace judgment in complex domains. It makes judgment harder to fake. It forces strategy to be evaluated against structured uncertainty rather than persuasive narrative. It turns "sounds right" into "survives these checks under these assumptions."

Chapter 6: The Convergence Thesis

Why Verification Requires Understanding

The parameterized simulation checks strategies against a model of the world built from historical patterns and expert-encoded structure. It surfaces assumptions, quantifies sensitivity, and stress-tests recommendations against documented uncertainties. Valuable work. Far better than unexamined narrative persuasion. But the structural assumptions themselves remain unvalidated by the simulation process. The causal arrows in the sketch, the categories used for calibration, the behavioral rules encoded for competitors: these are inputs, not outputs. The simulation explores variations within the encoded structure. It does not check whether the structure tracks reality.

The stakes are real. Billions in capital allocation depend on assumptions that simulations cannot validate. Market entry strategies rest on competitive response models that encode how rivals behaved historically, not how they will behave when circumstances change. Policy recommendations propagate through organizations on the strength of stress tests that explore variations within a causal sketch while leaving the sketch itself unexamined. The gap between "explored variations within this structure" and "verified this structure tracks reality" is where catastrophic errors live. Organizations that mistake the former for the latter will eventually encounter a scenario where the structure was wrong in ways the simulation could not surface.

Contrast this with what a skilled analyst firm would do when retained to evaluate the South American market entry.

The analyst firm investigates rather than projects. It interviews regional competitors to understand their strategic priorities, their capacity constraints, their relationships with regulators. It maps distribution channels through conversations with logistics operators, retailers, and former employees of target partners. It assesses the regulatory environment by understanding which officials make decisions, what political pressures they face, and how policy has actually evolved in response to past lobbying. It senses the texture of the market through primary research that no dataset captures.

The analyst firm synthesizes across sources in ways that no historical calibration can replicate. It notices that the company's strongest potential partner has a new CEO with a track record of aggressive renegotiation. That information appears nowhere in the structured data but dominates the risk profile. It recognizes that a recent trade agreement creates options that did not exist in the historical precedent period. It understands that the competitive response rules encoded in the simulation miss how one specific competitor thinks about market share, because an analyst spent two days with their former head of strategy.

The distinction is not about data quality. The analyst firm gathers information that does not exist in any dataset. It synthesizes that information through contextual understanding that pattern-matching cannot replicate. It makes judgments about what questions matter, what signals are reliable, and what assumptions deserve scrutiny. It brings understanding to the domain in a way that enables genuine verification rather than stress-testing under fixed structural assumptions.

What would it take to close this gap? What capabilities would enable verification that the structural assumptions themselves are correct? The answer emerges from examining why simpler approaches fail and what actually works when verification succeeds.

The Statistical Barrier

The most natural response to verification challenges is more data. Train a verifier on enough examples and the statistical patterns will generalize. If the generator produces outputs and the verifier learns to distinguish good from bad, perhaps scale solves the problem. This intuition powers much of modern machine learning. It does not survive contact with the mathematics of causation.

Judea Pearl's causal hierarchy provides the framework. The hierarchy distinguishes three levels: association, which captures what typically happens together; intervention, which captures what happens if I act; and counterfactual, which captures what would have happened if I had acted differently. Language models operate convincingly at level one. They excel at associational patterns. They can recall that smoking correlates with cancer. But they fail systematically at levels two and three. They cannot reliably predict the downstream effects of a novel intervention in a system they have not seen.

The Causal Hierarchy Theorem, proven by Bareinboim and colleagues, establishes why.[23] Data at one layer virtually always underdetermines information at higher layers. To answer questions at level two, you need knowledge at level two or higher. Statistical models at level one cannot answer interventional questions at level two, and neither can answer counterfactual questions at level three, without additional causal assumptions. No amount of observational data bridges these layers. No dataset size substitutes for structural knowledge about how the domain works.

Verification of a complex plan is inherently a level three task. To verify the claim "This strategy is sound," you must effectively ask: would the outcome have been disastrous if the environment were slightly different? Would competitors have responded differently under plausible alternative conditions? Would the regulatory environment have shifted in ways that invalidate the assumptions? A strategic guarantee is a claim about unobserved worlds. The verifier must possess a causal model of the domain. Relying only on statistical checks leaves the verifier mathematically incapable of verifying causal validity.

Pearl summarized this limitation: all the impressive achievements of deep learning amount to just curve fitting.[22] The curves may fit observed data beautifully while missing the causal structure entirely. A verifier trained on historical examples of successful and unsuccessful market entries learns correlations between features and outcomes. It cannot determine whether the causal mechanisms that produced those historical outcomes still operate, whether they apply to the specific context under evaluation, or whether novel factors will dominate that appeared in no training example.

Yann LeCun's JEPA architecture represents one attempt to address this limitation.[24] JEPA learns by predicting representations in abstract embedding space rather than pixel or token space, building internal models of domain dynamics rather than surface statistics. Yoshua Bengio's work on System 2 deep learning makes a parallel argument[25]: current models excel at System 1 tasks like pattern matching but fail at System 2 tasks requiring abstract reasoning, planning, and causal understanding. Verification of causal claims requires causal reasoning. Architectures that lack this capacity cannot acquire it through scale alone.

The statistical barrier is absolute within its scope. Pattern recognition over historical data cannot verify claims about how interventions will unfold in novel contexts. But perhaps verification need not reach the causal level. Perhaps behavioral correctness suffices. If outputs pass enough tests, match enough criteria, satisfy enough constraints, does it matter what process generated them?

The Behavioral Barrier

We verify pharmaceuticals by running clinical trials, not by understanding every molecular interaction. We verify bridges by load testing, not by simulating every stress distribution. Behavioral correctness, observable outcomes matching specifications, might suffice for AI systems as well. Understand the inputs, check the outputs, treat the middle as irrelevant.

The philosophical problem with this approach is not practical but logical.

You watch a child learning addition solve 2+3=5, then 4+1=5, then 7+2=9. Has the child learned addition? The problem, articulated by Wittgenstein and sharpened by Kripke[26], is that any finite set of examples is compatible with infinitely many rules. Perhaps the child has learned addition. Or perhaps the child has learned a deviant rule that coincides with addition for small numbers but diverges for larger ones. Call it quaddition. No finite behavioral check can distinguish which rule the child actually internalized.

Kripke made this vivid with the plus/quus example. Suppose you have never added numbers greater than 57. You answer 68+57=125, believing this accords with "plus." But all your past behavior is consistent with "quus," where x quus y equals x+y if both are less than 57, else equals 5. No fact about your mental states determines you meant plus rather than quus. A verifier checking the answer cannot determine which function the system followed.

The puzzle is not a fanciful exercise for philosophy seminars. It describes exactly the situation with AI verification: all training data is finite, all evaluation is finite, and infinitely many functions are consistent with any finite set of examples. Verification of a novel case requires knowing which function was intended. Knowing which function was intended requires understanding what the system is actually doing, not just what outputs it produces.

The Gettier problem reveals a further complication.[34] A verification system can produce true verifications while failing to constitute knowledge, because the verification may be correct for reasons disconnected from the system's justification. A model might verify that a legal citation is accurate because it pattern-matches to citations seen in training, not because it understands what makes a citation valid. When the pattern holds, verification succeeds. When the pattern breaks, no mechanism exists to detect the failure. Correct output does not imply correct reasoning. The next input may trigger the divergence.

Loru et al.[33] coined the term "Epistemia" to describe the epistemic condition created by generative AI: a structural situation in which linguistic plausibility substitutes for epistemic evaluation. The outputs appear increasingly human-like, yet the underlying processes remain misaligned with truth-tracking. Because language model output is path completion rather than belief formation, verifying it requires stepping out of the linguistic loop and into the evidential loop. Relying on fluency means succumbing to Epistemia, not verifying truth.

Behavioral verification cannot determine which rule a system follows. Correct outputs may arise from processes that will fail on the next input. The barrier is logical, not merely practical. Together with the statistical barrier, it eliminates the easy paths to verification. Patterns over historical data cannot verify causal claims. Observable outputs cannot determine underlying process. What, then, enables verification when it actually succeeds?

The World Model Requirement

A friend shows you architectural plans for a building addition and asks: will this structure be stable? If you are not an engineer, you cannot verify the claim. You can check that the drawing looks professional. You can confirm the architect has credentials. But you cannot assess whether the load-bearing calculations are correct because you lack the ability to simulate how forces propagate through the structure. You do not have the world model. Without it, verification is impossible.

Now consider when verification works.

A domain expert reviews the parameterized simulation for the South American market entry. The harness has exposed every structural assumption: how demand

responds to pricing, how competitors react to entry, how tariffs affect margin, what distributions calibrate each parameter. The expert studies these assumptions. She recognizes that the competitive response model assumes rational profit-maximization, but she knows the leading regional competitor is family-owned and optimizes for market share to preserve legacy. She flags the assumption. The model updates. This is successful verification.

What separates the expert from the architectural novice? The expert possesses a mental model of the domain. She can simulate how competitors actually behave, not just how a rational-actor framework predicts they will behave. When she encounters an assumption, she runs it against her internal model and checks for fit. The verification is simulation, not pattern-matching against historical examples or behavioral checking of outputs against test cases. She imagines the scenario unfolding under the stated assumptions and notices where her understanding of the domain diverges from the encoded structure.

Cognitive psychology has documented this mechanism extensively. Philip Johnson-Laird's mental model theory, developed since 1983[27], demonstrates that reasoning is simulation-based rather than rule-based. People infer that a description is consistent only if they can construct a model in which all assertions are true. Verifying a claim requires building a mental model of the scenario and checking whether the claim holds within that model. The expert verifies the competitive response assumption by constructing a mental model of the competitor, simulating their likely response, and comparing it to what the assumption predicts.

Lawrence Barsalou's perceptual symbol systems theory provides the neurological mechanism.[28] Comprehension involves reactivating sensory-motor representations captured during perceptual experience. The expert who spent years in the industry has accumulated representations of how markets work, how competitors think, how regulatory environments evolve. When she reads an assumption, she reactivates those representations and runs the assumption against them. The verification is a render, not a logic check.

The illusion of explanatory depth makes the absence of such models dangerous. Rozenblit and Keil documented across twelve studies that people systematically overestimate their understanding of causal mechanisms. Understanding ratings

dropped sharply after participants attempted to explain how devices work.[29] People who believe they can verify often cannot. The feeling of understanding is not understanding. Only the actual capacity to simulate the domain enables verification.

Newton's (1990) classic experiment captures the dynamic with clarity.[30] Researchers asked "tappers" to tap out well-known songs and predict how often listeners would identify them. Tappers predicted 50% success. Actual identification rates were 2.5%. The tappers heard the symphony in their heads; the listeners heard only Morse code. The expert verifying market entry assumptions hears the symphony of competitive dynamics, regulatory evolution, and market structure. Someone without that expertise hears only the taps of encoded parameters. Without the internal model, without the domain playing in your head, verification produces confident judgments that track nothing.

The pattern is now clear. Verification succeeds when the verifier possesses a world model of the domain sufficient to simulate the claims being checked. Statistical patterns cannot substitute for this model because they operate at the associational level while verification requires causal reasoning. Behavioral checks cannot substitute because they test outputs rather than the processes that generate them. What verification requires is the capacity to model how the domain actually works and simulate outcomes under varying assumptions.

This explains why parameterized simulation paired with domain expertise works. The simulation exposes assumptions in a form experts can evaluate. The experts bring world models accumulated through years of experience. Together they achieve verification that neither could accomplish alone. The simulation surfaces structure; the expert checks that structure against reality.

The Convergence

The verification gap and the AGI gap are the same gap.

Statistical patterns fail because verification requires causal reasoning that associational data cannot provide. Behavioral checks fail because finite outputs are compatible with infinitely many underlying processes. What actually enables verification is the capacity to model how domains work and simulate outcomes under varying conditions. The domain expert succeeds not because she has seen more

examples but because she carries a world model that supports counterfactual reasoning about the domain. She can ask: what would happen if this assumption were wrong? And she can simulate the answer.

Building systems that possess this capacity for arbitrary domains is exactly what we mean by artificial general intelligence. Active investigation: gathering information no dataset contains. Contextual synthesis: connecting signals across sources in ways pattern-matching cannot replicate. Causal understanding: modeling how domains actually work rather than how historical patterns suggest they might work. The capabilities required to verify complex claims about open-world outcomes are the capabilities that would constitute general intelligence.

The implication reframes the trajectory of AI development. When we say that current AI cannot reliably verify complex outputs in open domains, we are saying it lacks the world models and active investigation capacity that genuine understanding requires. When we say current AI is not general intelligence, we are saying the same thing in different words. Closing one gap closes the other.

The harness architecture clarifies the scope of this conclusion rather than contradicting it. For formal domains where heterogeneous verification applies, where external systems provide ground truth independent of the generator's reasoning, the harness can deliver reliability today. For parameterized simulation paired with domain expertise, the harness delivers structured uncertainty under explicit assumptions that experts can evaluate. The architecture succeeds where world models already exist, either encoded in formal systems or carried in expert minds.

Where those world models do not exist, where the domain is novel or the expertise is unavailable or the complexity exceeds human simulation capacity, the verification gap remains open. Closing it requires building the world models themselves. That project and the project of building artificial general intelligence are the same project.

Implications for the Harness

The harness serves as transitional infrastructure for the era between statistical pattern matching and genuine understanding.

The capabilities the harness forces into existence are the capabilities that future foundation models will need to internalize. Orchestration for exploring possibility

space. Preference learning for capturing human judgment. Parameterized simulation for stress-testing against structured uncertainty. What is external scaffolding today becomes native architecture tomorrow. The harness trains the model to confront reality in ways that mere next-token prediction cannot.

If this trajectory continues, the organizations that build the most capable verification systems will have built the foundations for AGI. They will have done so not by training larger models than their competitors, but by forcing models to confront reality through verification that cannot be gamed, grounding that cannot be faked, and feedback that accumulates into understanding.

The route to AGI may well run through verification rather than generation. We build systems that can check outputs against reality, and the checking requires such thorough world modeling that the verification system eventually possesses the understanding that would enable it to generate what it was built to verify. The harness becomes the intelligence. Not metaphorically but architecturally.

World models are the substrate on which verification of open-world outcomes depends. Building them is the direct route to both reliability and intelligence. The convergence thesis does not predict when this route will be traveled or who will travel it first. It identifies the destination and establishes that no shortcut leads there.

Part III: How Value Accumulates

Chapter 7: The Development Trajectory

Phased Projections for Capability Emergence

The preceding chapters established what the harness is and why it matters. What remains is when and that question admits only probabilistic answers. Each layer of the harness reaches production maturity at different rates. Some capabilities exist today. Others remain years away. The projections below represent plausible scenarios rather than confident predictions. Technology forecasting rewards humility.

Phase 1: Foundation Loops (2025-2026)

The foundation layer works now. Poetiq demonstrated that refinement loops succeed for complex reasoning tasks. Automated verification for formal domains achieves reliability approaching 95% for code, mathematics, and compliance. AlphaProof demonstrated that neural-symbolic integration can solve problems at the frontier of human capability. AlphaCodium showed that sophisticated iteration can more than double coding accuracy with minimal model invocations. The harness infrastructure for foundation-layer work exists; integration work is substantial but well-characterized.

Over the next twelve to eighteen months, these capabilities become standard organizational infrastructure. Organizations will deploy AI systems that iterate automatically against their specific quality checks, particularly for code generation, document compliance, and structured data transformations. Constant supervision becomes unnecessary when verification is objective. The limiting factor is integration: connecting harness systems to enterprise data, workflows, and quality criteria.

Phase 2: Domain Loops (2026-2028)

Semi-automated verification batteries for structured domains will mature during this phase. Evidence grounding combined with adversarial testing achieves 85-90% reliability on legal analysis, medical synthesis, and scientific domains. The systems cannot guarantee correctness, but they can flag inconsistencies, verify citations, and stress-test reasoning before human review.

Human oversight remains valuable but becomes supervisory rather than line-by-line. A legal associate reviewing AI-drafted briefs focuses on strategic judgment and edge cases rather than citation accuracy, which the system has already verified. A clinical researcher reviewing AI synthesis focuses on interpretation and implications rather than whether the underlying studies support the claims, which the system has already checked. The human role shifts from quality control to quality direction.

Phase 3: Differentiation Loops (2026-2029)

Continuous learning from implicit signals becomes standard during this phase. Personal adapters capture judgment across organizations. The differentiation layer, where taste and strategy live, finally becomes tractable. The technical components exist in research today. The integration and productization work spans this period.

When preference loops close, generic AI becomes personal AI. The system that drafted adequate marketing copy now drafts marketing copy in your voice, reflecting your accumulated judgment about what works. The strategic analysis tool that produced generic recommendations now produces recommendations weighted by your risk tolerance and strategic priorities. This is when enterprise AI adoption accelerates dramatically, because the systems finally learn what specific users and organizations actually want.

Phase 4: World Model Integration (2027-2030+)

Causal simulation for complex verification emerges during this phase. Calibrated confidence replaces binary verdicts across domains. The convergence between sophisticated verification and general intelligence becomes apparent. This is the AGI threshold: not necessarily human-level performance across all tasks, but systems with sufficient world models to verify complex claims about unfamiliar domains.

The compute wall may constrain this phase. Projections from Epoch AI suggest training costs exceeding $1 billion by 2027[31], with energy requirements approaching city-scale. If algorithmic efficiency does not improve dramatically, the brute-force path to AGI may stall. The harness thesis offers a partial hedge: if model scaling saturates, harness sophistication becomes the primary vector for capability improvement. The two axes provide diversified exposure to different technical trajectories.

Uncertainty Acknowledgment

These projections assume continued progress without discontinuous breakthroughs or setbacks. Major algorithmic innovations could accelerate timelines significantly; regulatory interventions, compute constraints, or safety incidents could delay them. The projections are scenarios, not predictions: internally consistent stories about how capabilities might develop given current trajectories. Wise organizations plan for uncertainty. Specific timelines will prove wrong. The direction of travel will not.

Chapter 8: Strategic Implications

For Leaders, Investors, and Researchers

The harness thesis carries distinct implications for different stakeholders. What follows translates the architectural framework into strategic priorities for those making decisions about AI investment, deployment, and research.

For Enterprise Leaders

The harness thesis reorders what matters for enterprise AI deployment. The limiting factor is no longer model capability but verification infrastructure. Organizations that recognize this shift can act on it by identifying where in their operations correctness can be verified programmatically. Consider where code compiles against test suites, where compliance checks run against codified rules, where data transformations validate against schemas. These domains are candidates for full automation today, not because the models have become perfect, but because the harness can catch their errors. The technology exists. The integration work remains.

Beyond these immediate opportunities lies a deeper investment: domain formalization. Every constraint that can be made explicit becomes verifiable with high reliability. The legal team that codifies its citation standards enables automated verification of citation accuracy. The medical practice that encodes its clinical guidelines enables automated conformance checking. This is not mere documentation. It is infrastructure. Formalization is the prerequisite for automation, and the organizations that invest in it now will compound that advantage as harness capabilities mature.

The preference gap presents a different kind of opportunity. Every time an expert chooses between alternatives, that choice encodes judgment. Which draft did the senior partner prefer? Which concept did the creative director select? Which recommendation did the executive implement? These signals are currently lost. Organizations that systematically capture them will have compounding advantages when preference learning matures. The infrastructure need not be elaborate (selection tracking, A/B frameworks, structured feedback loops) but it must exist before the judgment can be learned.

Finally, the harness architecture clarifies where humans remain essential. They validate world model assumptions. They adjudicate when verification systems disagree. They provide judgment on questions that resist formalization. The question is no longer whether to use AI but how to design collaboration protocols that combine machine reliability with human judgment. Organizations that answer this question well will deploy AI that actually works. Those that do not will join the 88% stuck in pilot purgatory.

For Investors

The harness thesis suggests a significant reallocation of value in the AI stack. Foundation model training has captured most investment attention, but the returns to additional training may be diminishing while returns to orchestration remain high. This suggests opportunity in the harness layer: companies building verification systems, preference learning infrastructure, and domain-specific orchestration.

The investment profile differs substantially from foundation model investment. Foundation models require massive upfront capital with binary outcomes. Harness development follows a different profile: iterative investment with continuous feedback and incremental improvement. This OpEx-dominant profile reduces risk and allows faster course correction, making harness ventures more amenable to traditional venture economics than the winner-take-all dynamics of foundation model development.

The commoditization dynamic matters. If foundation models become interchangeable substrates, as the frozen foundation principle suggests, then model providers face pricing pressure while harness providers capture differentiated value. The analogy to cloud computing is suggestive: compute became commodity while orchestration and application layers captured margin. A similar dynamic may play out in AI, with foundation models as the commodity layer and harnesses as the differentiation layer.

For Researchers

The convergence thesis identifies verification as a path to AGI that complements rather than competes with model scaling. Research that improves verification capabilities, whether through better world models, more sophisticated preference

learning, or more effective orchestration, contributes to the same ultimate goal as research that improves generation capabilities. The two paths converge.

The analysis points toward specific research frontiers. Causal reasoning remains the critical gap. Systems that model interventions and counterfactuals, not just correlations, would unlock verification for domains where current methods fail. Preference learning from implicit signals offers a path to capturing judgment that resists articulation. World models that maintain grounding under distribution shift, rather than merely generating coherent simulations, would extend verification to complex domains. Each of these directions advances both capability and reliability. The convergence thesis suggests they are the same direction.

The verification focus also has implications for safety research. If verification requires understanding, then building systems that can verify complex claims about safety is tantamount to building systems that understand safety. The alignment problem and the verification problem may be the same problem. Research that advances verification advances alignment, and vice versa.

Conclusion

The Path To Reliability

This paper began with a puzzle. A six-person startup achieved state-of-the-art on the most demanding reasoning benchmark by building better scaffolding around existing models. The same foundation models that struggled when prompted directly excelled when wrapped in sophisticated iteration and verification loops. The intelligence was in the harness.

From this observation, we diagnosed a structural limitation in current AI. The problem is not a fixed accuracy ceiling that models cannot break. The problem is that single-step accuracy compounds destructively over sequential decisions: 95% per step yields 36% success over twenty steps, and enterprise workflows routinely involve dozens or hundreds. Models cannot check their own blind spots, and homogeneous verification cannot ground outputs in external truth. The preference gap follows from the same root: 88% of organizations remain stuck in pilot purgatory because generic systems cannot learn specific judgment. They cannot verify what specific users actually want.

The framework we proposed responds to this diagnosis. The harness architecture separates generation from verification, using heterogeneous systems to catch errors the generator cannot see. An orchestrator governs iteration and stopping. Preference learning accumulates judgment from implicit signals. Formal verification achieves near-perfect reliability where outputs compile against specifications. Parameterized simulation stress-tests complex domains against structured uncertainty, surfacing assumptions that human expertise must validate. Each layer addresses a specific limitation identified in the diagnosis.

The convergence thesis emerged from examining where verification becomes hard. Cognitive psychology, epistemology, and causal inference converge on the same conclusion: robust verification of complex claims requires the same capabilities as generating those claims reliably. Statistical patterns cannot verify causal claims. Behavioral checks cannot determine underlying process. What enables verification is a world model sufficient to simulate outcomes under varying conditions. There is no

cheap verifier for outputs that require genuine understanding. The verification gap and the AGI gap are the same gap.

This convergence points toward a path. We build harnesses as transitional architecture for the era between statistical pattern matching and genuine understanding. The capabilities we develop for verification are the capabilities that constitute intelligence: world models, causal reasoning, active investigation. Perhaps the route to general intelligence runs through verification rather than generation. When those capabilities mature sufficiently, the distinction between model and harness dissolves. What required external scaffolding becomes native capability.

The destination is systems that need no harness because they understand what they generate. We build the harness because we need it now, and because building it advances us toward systems that will not.

Open Questions

Open Questions for the future

The harness architecture rests on assumptions. Some will prove wrong. What follows names them.

The Investment Profile Shift

Harness-centric development changes the capital structure of AI deployment. Foundation models require massive upfront capital expenditure, with training runs measured in hundreds of millions of dollars. These investments carry binary outcomes and long feedback cycles. A training run either produces a capable model or it does not. There is no partial credit. Harness development follows a different profile: iterative investment with continuous feedback, modular components that can be tested independently, and incremental improvement rather than step-function breakthroughs.

Organizations can begin with lightweight verification through linting, reference checking, and basic consistency validation, then progressively add sophistication as needs justify. A compliance adapter can be developed and deployed independently of a preference learning system. A refinement orchestrator can improve without touching verification batteries. This modularity reduces risk and allows faster course correction, advantages that matter as organizations navigate uncertainty about which AI capabilities will prove most valuable. The shift from CapEx-dominant to OpEx-dominant investment profiles may prove as significant as any technical advance in the harness itself.

The Formalization Bottleneck

Verification requires ground truth, and ground truth requires formalization. For domains where expert knowledge remains tacit, the limiting factor may be domain formalization rather than AI capability. Strategy, creative direction, and organizational dynamics resist encoding precisely because their practitioners cannot articulate what makes good judgment good. The timeline for closing verification loops in these domains depends on progress in knowledge engineering as much as machine learning.

Extracting expertise from human heads and encoding it in checkable form is slow and expensive work. Paradoxically, AI cannot easily accelerate this process because the knowledge that needs extracting is precisely the knowledge that current systems lack. The formalization bottleneck may bind more tightly than the technical analysis suggests. Some domains will remain verification-resistant not because we lack capable models but because we lack formal specifications of what correct looks like.

World Model Maturity

The convergence thesis posits that sophisticated verification and general intelligence require similar capabilities. Building systems that can verify complex strategic recommendations means building systems that can reason about consequences, model alternatives, and simulate counterfactuals. The architectural direction is sound. The timeline remains genuinely unknown.

World models capable of grounding verification in complex domains may require breakthroughs we cannot currently anticipate. The gap between Genie 3's impressive but physics-approximate simulations and the calibrated causal models needed for enterprise verification may close gradually or may require conceptual advances that resist scheduling. If such breakthroughs prove necessary, the verification gap for complex domains will persist longer than any phase diagram suggests. The destination is visible. The distance to it is not.

Organizational Adaptation

Verification-heavy workflows represent a significant departure from current interaction expectations. Users expect instant answers. Verification takes time. A query that returns in seconds with a raw model may require minutes with full verification, and the reliability gains that justify this latency are invisible to users who never see the errors that were caught.

Whether organizations will accept slower responses in exchange for higher reliability remains an open design challenge. The interface patterns that make this tradeoff palatable have not yet been developed. Progress bars and confidence indicators help, but the fundamental tension between immediacy and accuracy requires cultural as much as technical solutions. The technology may mature faster than the

organizational capacity to absorb it. Resistance to verification-induced latency could prove a more binding constraint than any architectural limitation.

The Internalization Question

The harness architecture treats foundation models as frozen substrates, building intelligence in the scaffolding around them. Yet this separation is likely temporary. The pattern has precedent: chain-of-thought prompting began as an external technique before being trained into models natively. Refinement loops that once required explicit orchestration now occur within extended thinking windows. Each capability that proves valuable at the harness layer becomes a candidate for absorption into the next generation of foundation models.

If this pattern continues, the harness layer may function less as permanent infrastructure and more as a proving ground for capabilities that eventually migrate inward. The strategic question shifts accordingly. Organizations investing in harness development must consider whether they are building durable competitive advantage or training data for the next model generation. The answer likely varies by layer: formal verification against external specifications cannot be internalized because it depends on ground truth that lives outside the model. Preference learning that encodes specific organizational judgment similarly resists absorption into general-purpose systems. But refinement orchestration, self-critique patterns, and verification heuristics may prove transient. The architectural direction remains sound even if specific implementations have shorter shelf lives than the thesis implies.

Economic Viability

Sophisticated harnesses carry real costs. Research on multi-agent systems reveals that achieving reliability improvements through orchestration can consume fifteen times more tokens than single-pass generation[19]. Each verification step, each refinement cycle, each critique loop translates directly into API calls and compute charges. A task that costs pennies in raw generation may cost dollars when wrapped in production-grade verification.

These economics create deployment constraints the architecture must accommodate. High-stakes decisions where error costs are severe can justify substantial verification

overhead. A legal analysis that prevents a multi-million dollar liability is worth considerable compute. A medical synthesis that avoids a misdiagnosis justifies the expense. But routine tasks cannot bear such costs. The harness architecture implies a tiered deployment model where verification intensity scales with consequence severity. Organizations must develop explicit policies mapping task categories to appropriate verification levels, accepting that full harness deployment remains economically viable only for decisions where reliability justifies the investment. The ceiling on harness sophistication is often financial rather than technical.

Adapter Composability

The interference patterns that emerge when multiple adapters modify the same activation spaces, the debugging complexity of systems where errors could originate in any of several stacked modules, and the performance degradation that accumulates across composition layers are active research areas without clear answers. Modular adapters offer theoretical advantages over monolithic fine-tuning, but the practical limits of stacking them remain to be discovered. The architecture assumes clean composition. Reality may prove messier than the diagrams suggest.

Calibration and Trust

The harness architecture outputs confidence scores rather than binary verdicts. A strategic recommendation arrives with a 72% confidence rating conditioned on specified assumptions. A compliance check returns not just pass or fail but a probability distribution over potential issues. These calibrated outputs are useful only if users trust them and interpret them correctly.

The organizational and interface design required to build appropriate reliance on probabilistic outputs remains largely unexplored territory. Users may anchor on point estimates and ignore uncertainty bands. They may distrust low confidence scores and override correct caution. They may trust high confidence scores and miss the rare but consequential errors that slip through. Getting the technology right is necessary but not sufficient. Getting the human factors right may prove equally difficult. The research base for calibrated trust lags far behind the technical capabilities the harness enables. That gap, too, must close.

References

[1] Poetiq. (2025). "ARC-AGI-2 SOTA at Half the Cost." poetiq.ai/posts/arcagi_verified.

[2] McKinsey & Company. (2025). "The State of AI in 2025." McKinsey Global Survey.

[3] Boston Consulting Group. (2025). "From Potential to Profit: Closing the AI Impact Gap." BCG AI Radar

[4] Gartner. (2024). "Predicts 2025: AI Agents Transform Business Operations." Gartner Research.

[5] ARC Prize Foundation. (2025). "ARC Prize 2025 Results and Analysis." arcprize.org.

[6] Zhang, Y., et al. (2025). "Exploring the role of large language models in the scientific method: from hypothesis to discovery." npj Artificial Intelligence.

[7] SimpleBench Consortium. (2025). "SimpleBench: A Benchmark for Basic Physical and Social Reasoning." simplebench.ai.

[8] Hassabis, D. (2025). "Gemini for the United Kingdom." Google Cloud Event.

[10] Huang, Y. & Yang, L.F. (2025). "Winning Gold at IMO 2025 with a Model-Agnostic Verification-and-Refinement Pipeline." arXiv:2507.15855.

[11] Luo, Z., Glavas, G., Stassen, L. M., & Vulić, I. (2023). "An Empirical Study of Catastrophic Forgetting in Large Language Models During Continual Fine-tuning." arXiv:2308.08747.

[12] Li, H., Ding, L., Fang, M., & Tao, D. (2024). "Revisiting Catastrophic Forgetting in Large Language Model Tuning." arXiv:2406.04836.

[13] Qi, X., et al. (2024). "Fine-tuning Aligned Language Models Compromises Safety." ICLR 2024.

[14] Snell, C., Lee, J., Xu, K., & Kumar, A. (2024). "Scaling LLM Test-Time Compute Optimally Can Be More Effective than Scaling Model Parameters." arXiv:2408.03314.

[15] Ridnik, T., Kredo, D., & Friedman, I. (2024). "Code Generation with AlphaCodium: From Prompt Engineering to Flow Engineering." arXiv:2401.08500.

[16] Poetiq. (2025). "ARC-AGI-2 SOTA at Half the Cost." poetiq.ai/posts/arcagi_verified. (Self-auditing module reference)

[17] Hubert, T., Mehta, R., Sartran, L., et al. (2025). "Olympiad-Level Formal Mathematical Reasoning with Reinforcement Learning." Nature.

[18] Huang, Y. & Yang, L.F. (2025). Ibid. (Same as [10])

[19] Anthropic. (2025). "Building Multi-Agent Systems." Anthropic Engineering Blog. (Token consumption statistics)

[20] Huang, J., et al. (2024). "Large Language Models Cannot Self-Correct Reasoning Yet." arXiv:2310.01798.

[21] Saad-Falcon, J., Buchanan, E.K., Chen, M.F., et al. (2025). "Weaver: Shrinking the Generation-Verification Gap with Weak Verifiers." Stanford University.

[22] Pearl, J. & Mackenzie, D. (2018). The Book of Why: The New Science of Cause and Effect. New York: Basic Books.

[23] Bareinboim, E., & Pearl, J. (2016). "Causal Inference and the Data-Fusion Problem." Proceedings of the National Academy of Sciences, 113(27), 7345-7352.

[24] LeCun, Y. (2022). "A Path Towards Autonomous Machine Intelligence." OpenReview.

[25] Bengio, Y. (2019). "From System 1 Deep Learning to System 2 Deep Learning." NeurIPS 2019 Keynote.

[26] Kripke, S. (1982). Wittgenstein on Rules and Private Language. Cambridge, MA: Harvard University Press.

[27] Johnson-Laird, P.N. (2010). "Mental Models and Human Reasoning." PNAS.

[28] Barsalou, L.W. (1999). "Perceptual Symbol Systems." Behavioral and Brain Sciences.

[29] Rozenblit, L. & Keil, F. (2002). "The Misunderstood Limits of Folk Science: An Illusion of Explanatory Depth." Cognitive Science.

[30] Newton, E.L. (1990). "Overconfidence in the Communication of Intent: Heard and Unheard Melodies." Doctoral dissertation, Stanford University.

[31] Cottier, B., et al. (2024). "How Much Does It Cost to Train Frontier AI Models?" epochai.org.

[32] Hoffmann, J., Borgeaud, S., Mensch, A., et al. (2022). "Training Compute-Optimal Large Language Models." arXiv:2203.15556.

[33] Loru, E., et al. (2025). "The simulation of judgment in LLMs." Proceedings of the National Academy of Sciences, 122(42), e2518443122.

[34] Gettier, E. L. (1963). "Is Justified True Belief Knowledge?" Analysis.

Disclaimer

The author used artificial intelligence tools during the preparation of this white paper. Specifically, Claude Opus 4.5 (Anthropic), Gemini 3 Deep Research (Google) and Kimi K2.5 (Moonshot AI) were used to assist with drafting and refining text. Gemini Nano Banana Pro (Google) was used to generate visualizations. The author reviewed and edited all AI-generated content and takes full responsibility for the accuracy, integrity, and originality of the final work.