Towards Trustworthy Autonomous Research

Wataru Kumagai
NexaScience / RIKEN TRIP-AGIS

AIRAS

AGNTCon + MCPCon Japan 2026

Key messages

1. **AI** now makes discoveries and **speeds up research**
2. **Human verification can't keep up** with this speed
3. **AIRAS** automates verification for **trustworthy research**

AIRAS

The Impact of AI on Research

Research breakthroughs with AI

Solve Open Problems

10 advances

on long-standing problems

GPT-6 Astra (OpenAI, 2026)

Reveal Hidden Phenomena

New physics formula

nonzero where zero was assumed

GPT-5.2 (OpenAI, 2026)

Discover New Mechanisms

New treatment targets

proposed & validated

Co-Scientist (Google, 2026)

Design New Materials

Tailored materials

designed & synthesized

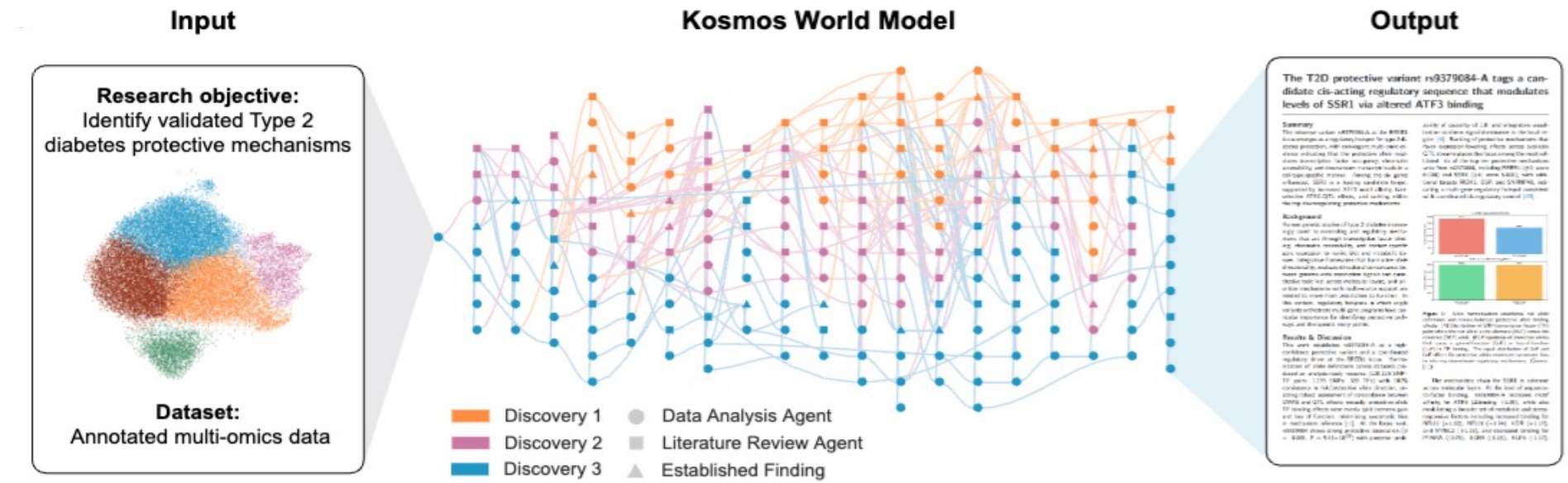
MatterGen (Microsoft Research, 2025)

Research breakthroughs by AI

Kosmos (Edison Scientific, 2025)

6 months → 1 day
(beta-user estimate)

4 novel contributions
across fields



Progress in AI Research Systems

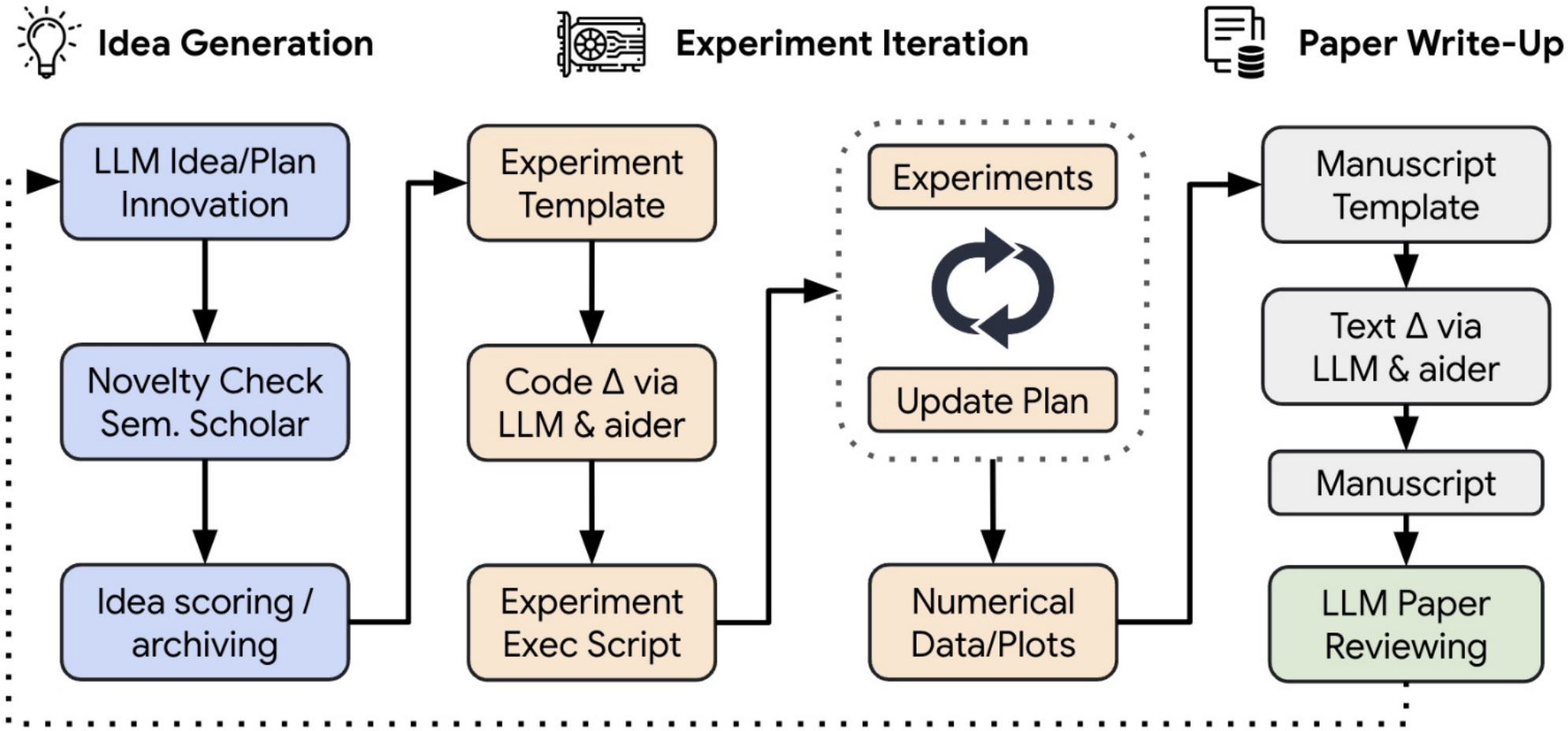
Research involves many steps

Until recently, researchers themselves handled all the steps.

Rise of conversational AI

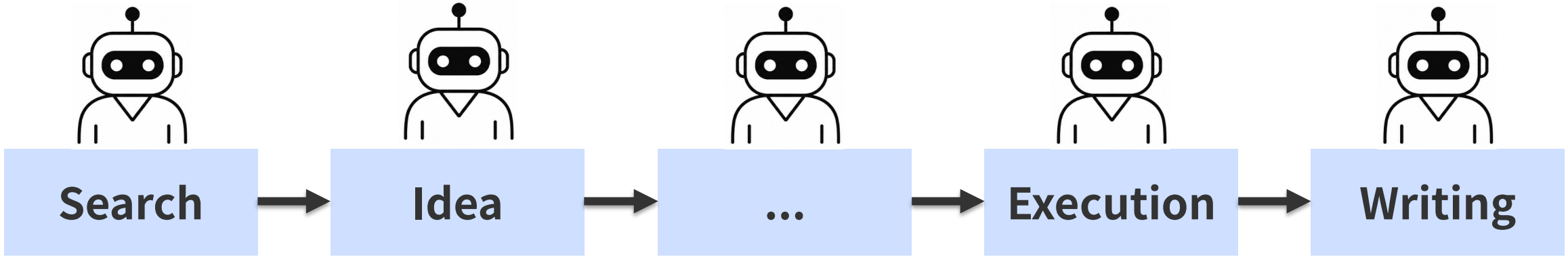
A single conversational AI can handle
many kinds of research tasks.

The AI Scientist and beyond



Lu et al., Nature, 2026

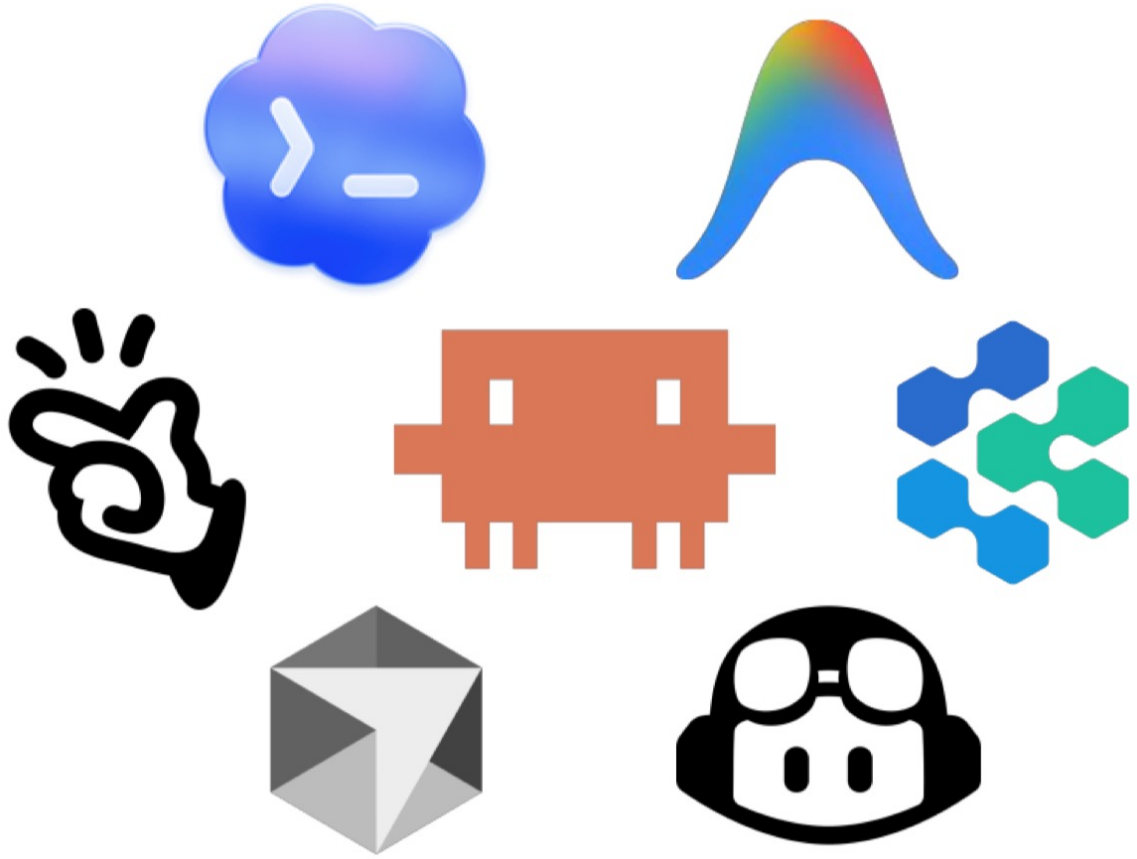
From manual work to workflow automation



A single system can handle all research steps end to end.

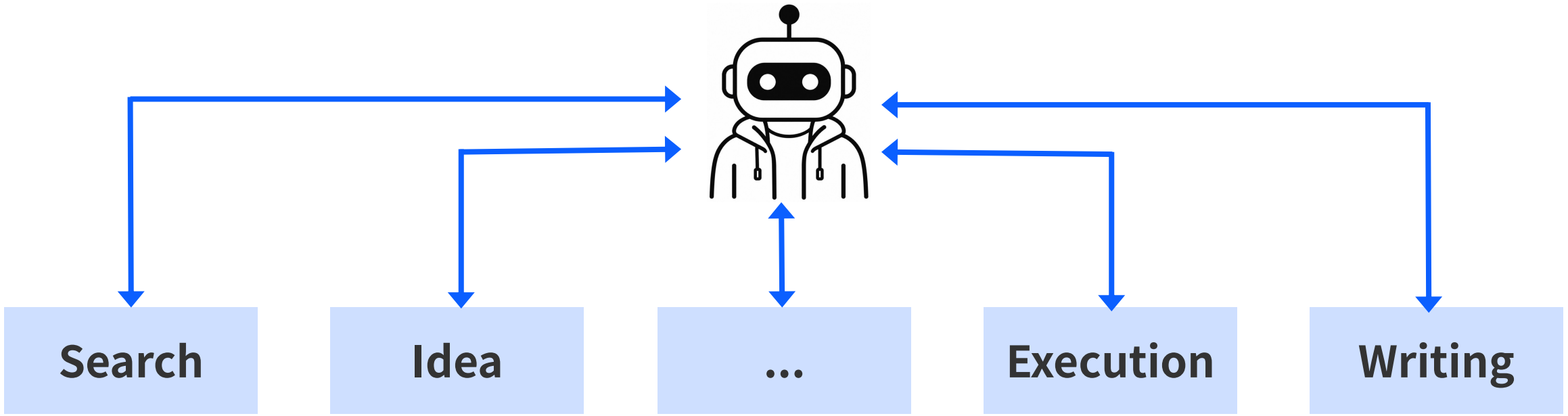
But the workflow is still fixed.

Evolution to coding agents



Coding agents **decide and act on their own**.

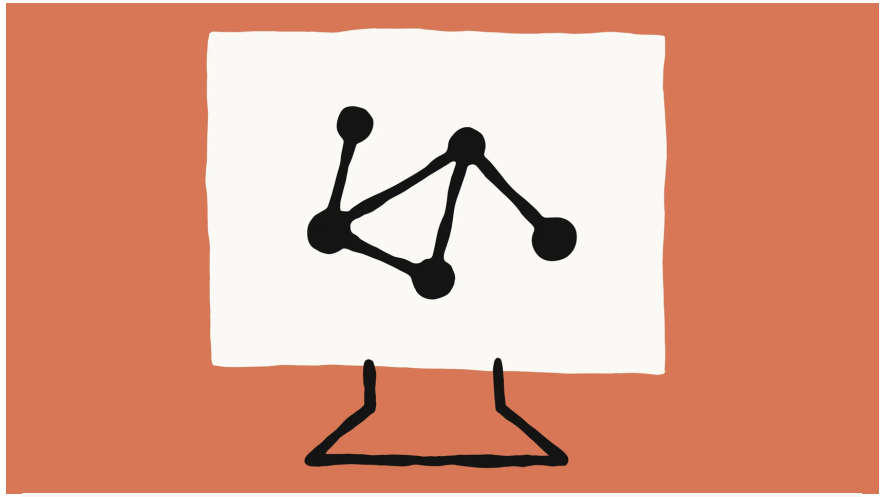
From fixed workflow to coding agents



Coding agents are flexible like human researchers.

But they are not specialized for research.

Specialized tools and environments for research



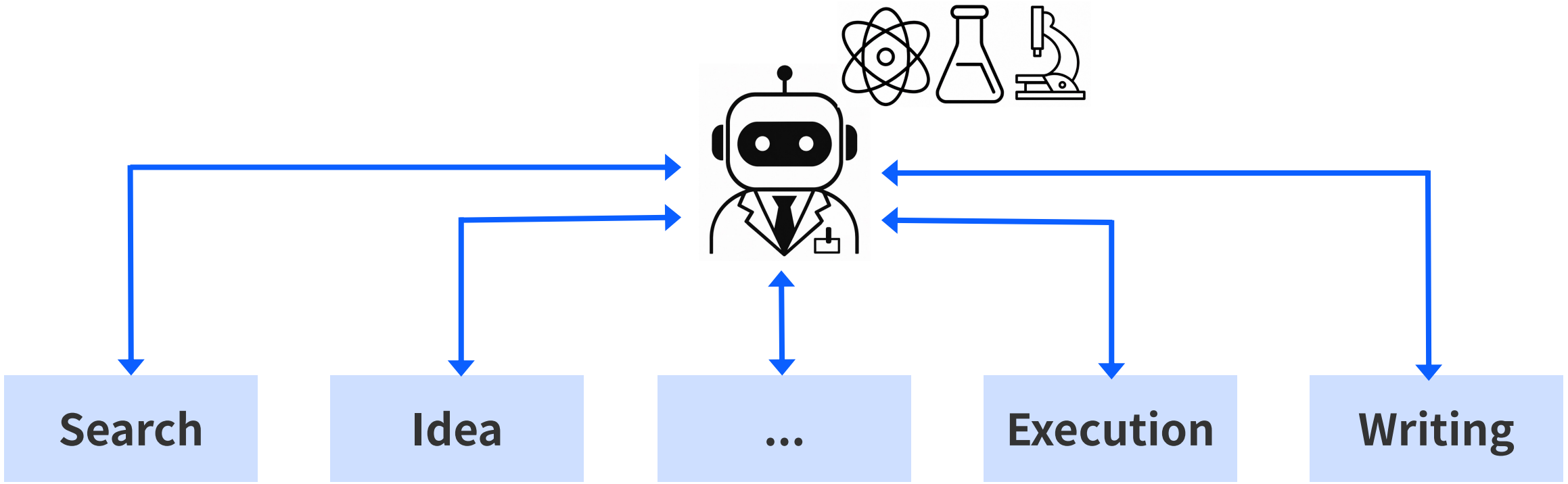
BioNeMo Agent Toolkit

Ai2 Asta resources

ToolUniverse

These environments and tools turn coding agents into research agents.

From coding agents to research agents



Research is becoming automated and faster.

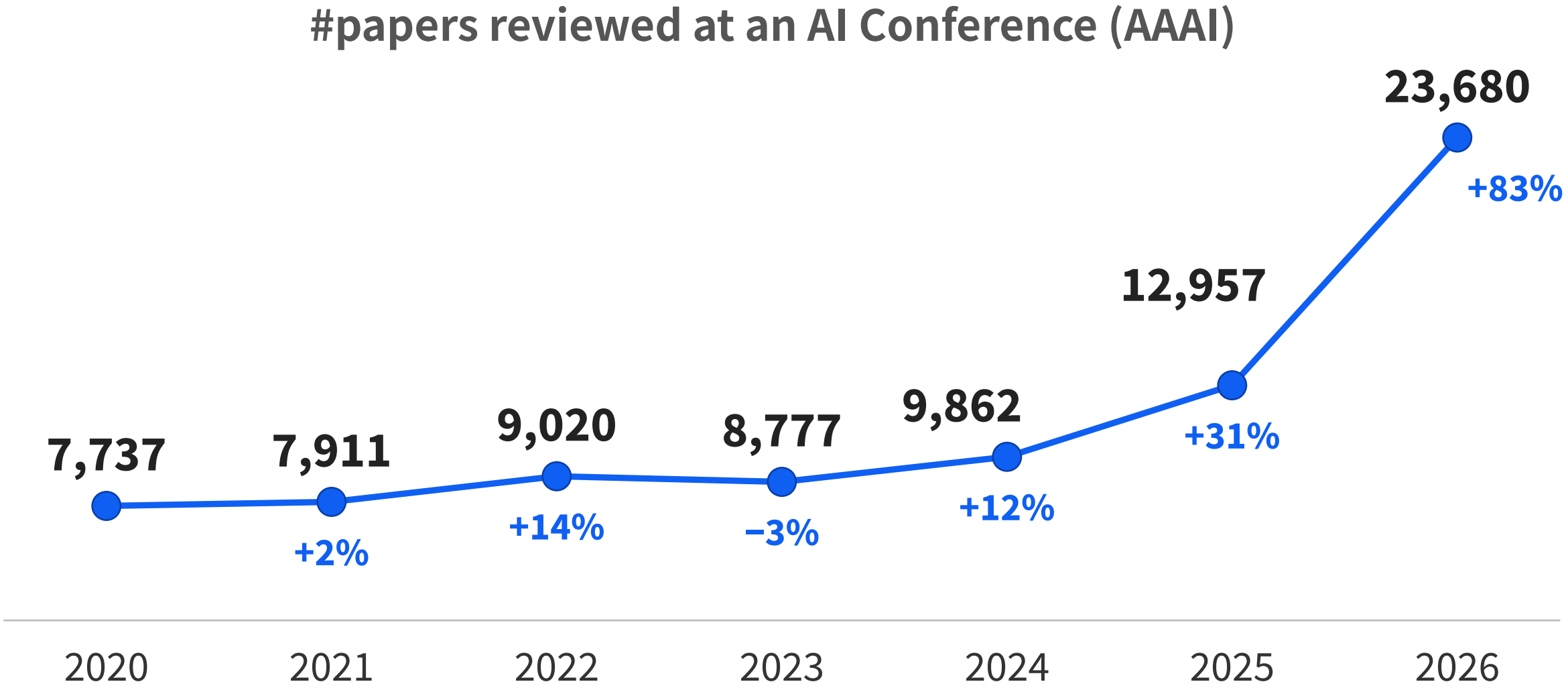
But faster research only helps if it can be trusted.

Peer Review for Trustworthy Research

Trust in research comes from peer review

Other researchers review every paper before it is published.

Accelerated paper submissions



Peer review is struggling

Surge in Peer Reviewers

Nearly 3x committee size

recruited vs. last year

AAAI 2026

Shortage of Experienced Reviewers

Undergraduate students

served as reviewers

ICLR 2025

AI-Generated Reviews

~21% of reviews

estimated fully AI-generated (Pangram)

ICLR 2026

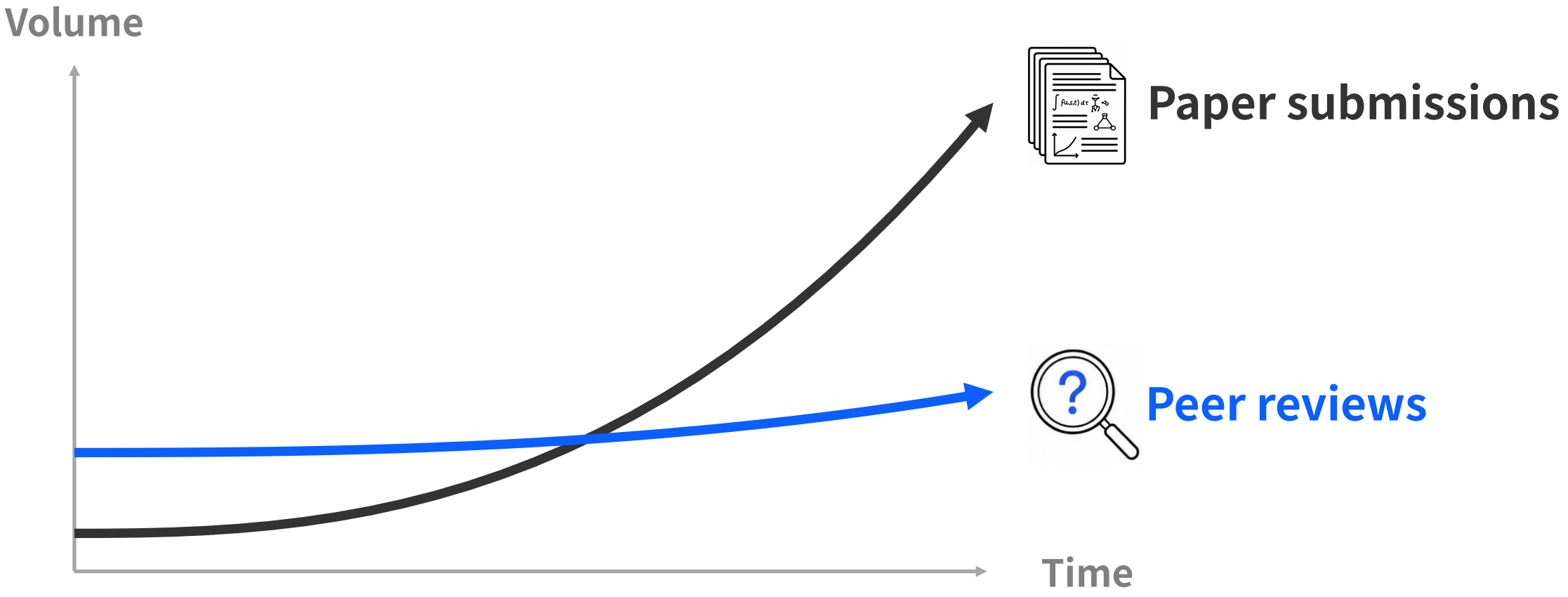
Manipulating AI-Assisted Reviews

Prompt injections

banned and screened by detectors

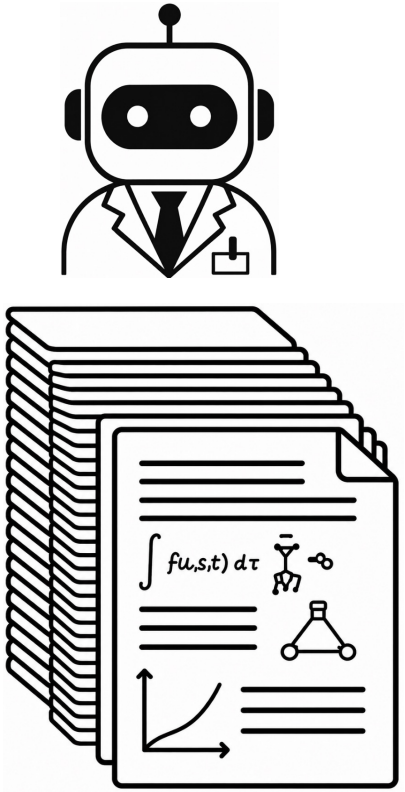
ICML 2026

The bottleneck has shifted to peer review



Research throughput ≤ min(Submissions, Reviews)

Agents produce faster than humans can check



Paper submissions

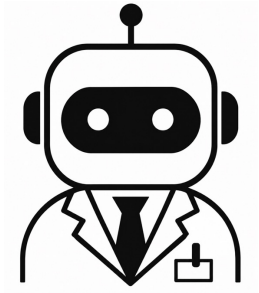
vs.

Peer review

How can we review papers in the age of AI research systems?

Reviewing AI-Generated Research

Results generated by AI research systems



Research System

Adaptive Memory Routing for Efficient Scientific Language Models

Elena Morita[1] elena.morita@nitx.ai
Julian Cross[2] julian.cross@rcmi.org
Nisha Verma[1] nisha.verma@nitx.ai
Theo Laurent[3] theo.laurent@hclab.dev
Mina Kovac[2] mina.kovac@rcmi.org

[1] Northbridge Institute of Technology [2] Redwood Center for Machine Intelligence [3] Helix Computational Lab

Abstract

Scientific language models must reason over long documents, tables, figures, and code while staying within practical memory budgets. We introduce Adaptive Memory Routing (AMR), a training and inference framework that dynamically allocates a token-level memory budget across multiple heterogeneous memory experts. AMR combines a lightweight router with a differentiable memory allocator that optimizes for task-aware information retention. During training, we minimize a composite objective that balances next-token prediction with a memory regularizer and a routing entropy term. At inference, AMR adaptively routes content to short-term, long-term, and external retrieval memories, yielding substantial efficiency gains with minimal quality loss. We evaluate AMR on five scientific reasoning benchmarks covering literature QA, math proofs, code synthesis, and table understanding. AMR consistently outperforms strong baselines, improving average task accuracy by up to 6.3% while reducing memory footprint by up to 37%. Analysis shows that AMR achieves superior performance by prioritizing evidence-bearing tokens and preserving cross-document dependencies. Extensive ablations validate the contributions of adaptive budgeting, router calibration, and expert diversity.

1 Introduction

Large language models (LLMs) have achieved impressive results across diverse scientific tasks, yet their quadratic memory cost limits context length and efficiency [1, 2]. Prior work addresses this via sparse attention [3], linear attention [4], or external retrieval [5], often trading off quality or flexibility. We propose Adaptive Memory Routing (AMR), which learns to allocate a fixed memory budget across multiple memory experts according to the content's utility for future prediction.

Given an input sequence $x_{1:T}$ with hidden states $h_t \in \mathbb{R}^d$, AMR uses a router $r_t \in \Delta^K$ over K memory experts. Each expert k maintains a memory M_k with capacity C_k and read function $\mathrm{Read}_k(\cdot)$. At time t, the router produces scores $s_{t,k}$ and assignment weights

$$r_{t,k} = \mathrm{softmax}_k\left(\frac{s_{t,k}}{\tau}\right), \quad \sum_{k=1}^{K} r_{t,k} = 1, \tag{1}$$

where τ is a temperature. Tokens are written using weighted updates $M_k \leftarrow \mathrm{Write}_k(M_k, h_t, r_{t,k})$ under expert-specific policies.

The model predicts the next token $\hat{y}_t$ via

$$p(\hat{y}_t | x_{1:t}) = \mathrm{softmax}(W_o\, g_t), \quad g_t = \mathrm{Merge}\left(\{\mathrm{Read}_k(M_k, q_t)\}_{k=1}^{K}\right), \tag{2}$$

with query $q_t = \phi(h_t)$ and learned merger Merge. We train with

$$\mathcal{L} = \mathcal{L}_{\mathrm{NLL}} + \lambda_1 \mathcal{L}_{\mathrm{mem}} + \lambda_2 \mathcal{L}_{\mathrm{ent}}, \tag{3}$$

where $\mathcal{L}_{\mathrm{NLL}}$ is the negative log-likelihood, $\mathcal{L}_{\mathrm{mem}}$ penalizes budget overflows via a soft capacity constraint, and $\mathcal{L}_{\mathrm{ent}} = -\sum_t \sum_k r_{t,k} \log r_{t,k}$ encourages confident routing early in training and is annealed later.

2 Method

AMR employs three heterogeneous experts: (i) Short-term memory (STM) with a FIFO buffer for local recency, (ii) Long-term memory (LTM) using a compressed key-value store with eviction by learned importance, and (iii) External retrieval (RET) backed by a vector index for cross-document facts. The router is a two-layer MLP with gated residual con
towards a u

Figure 1: Overview of Adaptive Memory Routing (AMR). The router dynamically allocates each token to one of three memory experts based on its estimated utility. Memories are read and merged to condition the decoder for next-token prediction.

3 Experiments

We evaluate AMR on five benchmarks spanning scientific reasoning, table understanding, and code generation (Table 1). Models are trained on a 1.2T-token mixture of scientific papers, preprints, textbooks, and code repositories. Baselines include a dense Transformer (Dense), Longformer [3], Retentive Transformer [6], and RAG-Style [5]. All models are matched in parameter count (7B) and trained with a context length of 32K tokens. Metrics are accuracy (↑) or exact-match (EM ↑).

Table 1: Performance comparison on scientific benchmarks.

Model	SciQA (EM ↑)	ProofBench (EM ↑)	TableQA (Acc ↑)	CodeGen (Pass@1 ↑)	LitQA (Acc ↑)	Avg (↑)
Dense (7B)	64.2	35.6	78.1	46.3	72.4	59.3
Longformer	66.8	38.9	79.6	49.7	74.8	61.9
Retentive	68.3	40.7	81.2	51.8	76.8	63.8
RAG-Style	70.6	42.6	83.5	53.6	79.3	65.9
AMR (ours)	**72.9**	**45.3**	**86.6**	**56.4**	**81.8**	**68.6**

AMR achieves the best average performance (+2.7% over RAG-Style) while reducing peak memory usage by 29–37% across tasks. Ablations show that removing any expert degrades performance, with the largest drop when removing LTM (−3.1%).

4 Results

Figure 1 illustrates AMR's architecture. We find that AMR allocates more capacity to LTM for cross-document tasks (LitQA, SciQA) and to STM for code generation. Router entropy annealing improves stability and final accuracy by up to 1.4%. Case studies reveal that AMR retains causal chains and table schemas more reliably than baselines.

5 Conclusion

AMR introduces adaptive, task-aware memory allocation that improves efficiency and accuracy for scientific language modeling. Future work includes dynamic expert creation and multi-modal memory routing.

References

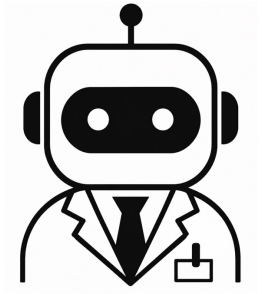
[1] A. Kim, J. Park, and S. Lee. Scaling laws of language model memory. *arXiv preprint arXiv:2401.01234*, 2024.
[2] M. Singh, J. Zhao, and Y. Wu. Efficient long-context modeling: A survey. *J. ML Research*, 25(7):1–45, 2024.
[3] Z. Beltagy, M. E. Peters, and A. Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Result

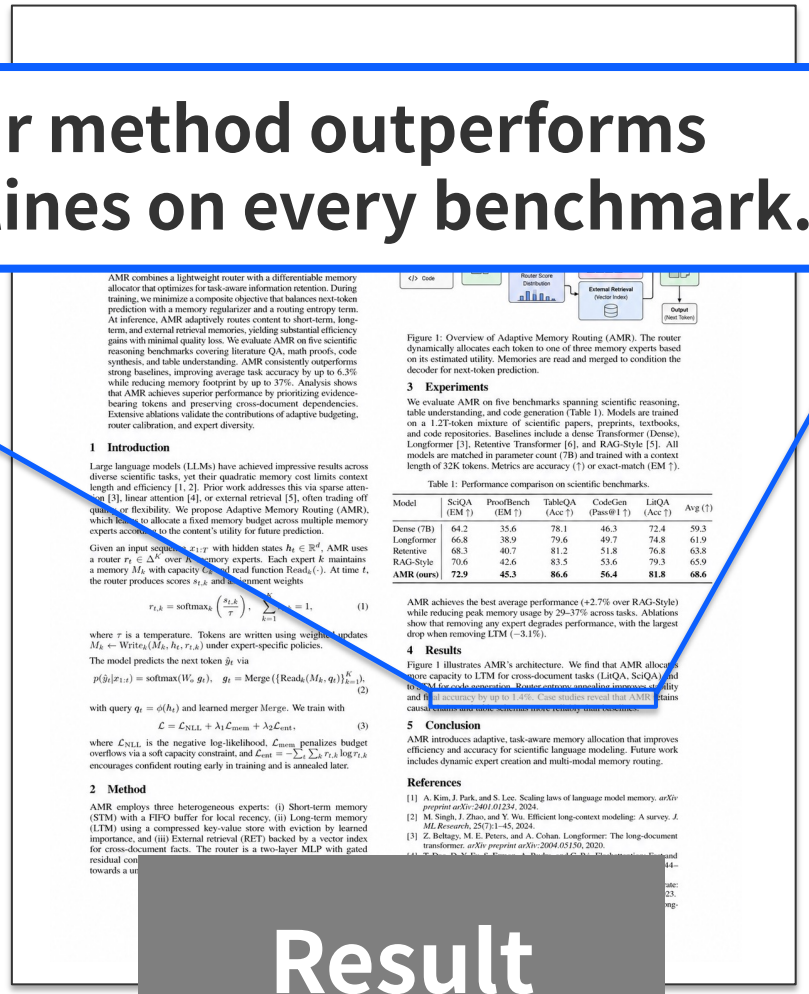
Results generated by AI research systems

Claim

"Our method outperforms all baselines on every benchmark."



Research System



AMR combines a lightweight router with a differentiable memory allocator that optimizes for task-aware information retention. During training, we minimize a composite objective that balances next-token prediction with a memory regularizer and a routing entropy term. At inference, AMR adaptively routes content to short-term, long-term, and external retrieval memories, yielding substantial efficiency gains with minimal quality loss. We evaluate AMR on five scientific reasoning benchmarks covering literature QA, math proofs, code synthesis, and table understanding. AMR consistently outperforms strong baselines, improving average task accuracy by up to 6.3% while reducing memory footprint by up to 37%. Analysis shows that AMR achieves superior performance by prioritizing evidence-bearing tokens and preserving cross-document dependencies. Extensive ablations validate the contributions of adaptive budgeting, router calibration, and expert diversity.

1 Introduction

Large language models (LLMs) have achieved impressive results across diverse scientific tasks, yet their quadratic memory cost limits context length and efficiency [1, 2]. Prior work addresses this via sparse attention [3], linear attention [4], or external retrieval [5], often trading off quality for flexibility. We propose Adaptive Memory Routing (AMR), which learns to allocate a fixed memory budget across multiple memory experts according to the content's utility for future prediction.

Given an input sequence $x_{1:T}$ with hidden states $h_t \in \mathbb{R}^d$, AMR uses a router $r_t \in \Delta^K$ over K memory experts. Each expert k maintains a memory M_k with capacity C_k and a read function $\mathrm{Read}_k(\cdot)$. At time t, the router produces scores $s_{t,k}$ and assignment weights

$$r_{t,k} = \mathrm{softmax}_k\left(\frac{s_{t,k}}{\tau}\right), \quad \sum_{k=1}^{K} r_{t,k} = 1, \tag{1}$$

where τ is a temperature. Tokens are written using weighted updates $M_k \leftarrow \mathrm{Write}_k(M_k, h_t, r_{t,k})$ under expert-specific policies.

The model predicts the next token $\hat{y}_t$ via

$$p(\hat{y}_t | x_{1:t}) = \mathrm{softmax}(W_o\, g_t), \quad g_t = \mathrm{Merge}\left(\{\mathrm{Read}_k(M_k, q_t)\}_{k=1}^{K}\right), \tag{2}$$

with query $q_t = \phi(h_t)$ and learned merger Merge. We train with

$$\mathcal{L} = \mathcal{L}_{\mathrm{NLL}} + \lambda_1 \mathcal{L}_{\mathrm{mem}} + \lambda_2 \mathcal{L}_{\mathrm{ent}}, \tag{3}$$

where $\mathcal{L}_{\mathrm{NLL}}$ is the negative log-likelihood, $\mathcal{L}_{\mathrm{mem}}$ penalizes budget overflows via a soft capacity constraint, and $\mathcal{L}_{\mathrm{ent}} = -\sum_t \sum_k r_{t,k} \log r_{t,k}$ encourages confident routing early in training and is annealed later.

2 Method

AMR employs three heterogeneous experts: (i) Short-term memory (STM) with a FIFO buffer for local recency, (ii) Long-term memory (LTM) using a compressed key-value store with eviction by learned importance, and (iii) External retrieval (RET) backed by a vector index for cross-document facts. The router is a two-layer MLP with gated residual con
towards a ur

Figure 1: Overview of Adaptive Memory Routing (AMR). The router dynamically allocates each token to one of three memory experts based on its estimated utility. Memories are read and merged to condition the decoder for next-token prediction.

3 Experiments

We evaluate AMR on five benchmarks spanning scientific reasoning, table understanding, and code generation (Table 1). Models are trained on a 1.2T-token mixture of scientific papers, preprints, textbooks, and code repositories. Baselines include a dense Transformer (Dense), Longformer [3], Retentive Transformer [6], and RAG-Style [5]. All models are matched in parameter count (7B) and trained with a context length of 32K tokens. Metrics are accuracy (↑) or exact-match (EM ↑).

Table 1: Performance comparison on scientific benchmarks.

Model	SciQA (EM ↑)	ProofBench (EM ↑)	TableQA (Acc ↑)	CodeGen (Pass@1 ↑)	LitQA (Acc ↑)	Avg (↑)
Dense (7B)	64.2	35.6	78.1	46.3	72.4	59.3
Longformer	66.8	38.9	79.6	49.7	74.8	61.9
Retentive	68.3	40.7	81.2	51.8	76.8	63.8
RAG-Style	70.6	42.6	83.5	53.6	79.3	65.9
AMR (ours)	**72.9**	**45.3**	**86.6**	**56.4**	**81.8**	**68.6**

AMR achieves the best average performance (+2.7% over RAG-Style) while reducing peak memory usage by 29–37% across tasks. Ablations show that removing any expert degrades performance, with the largest drop when removing LTM (−3.1%).

4 Results

Figure 1 illustrates AMR's architecture. We find that AMR allocates more capacity to LTM for cross-document tasks (LitQA, SciQA) and to STM for code generation. Router entropy annealing improves stability and final accuracy by up to 1.4%. Case studies reveal that AMR maintains causal chains and more selectively than fixed-size

5 Conclusion

AMR introduces adaptive, task-aware memory allocation that improves efficiency and accuracy for scientific language modeling. Future work includes dynamic expert creation and multi-modal memory routing.

References

[1] A. Kim, J. Park, and S. Lee. Scaling laws of language model memory. *arXiv preprint arXiv:2401.01234*, 2024.

[2] M. Singh, J. Zhao, and Y. Wu. Efficient long-context modeling: A survey. *J. ML Research*, 25(7):1–45, 2024.

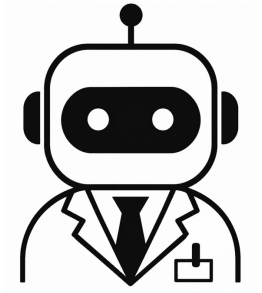
[3] Z. Beltagy, M. E. Peters, and A. Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Result

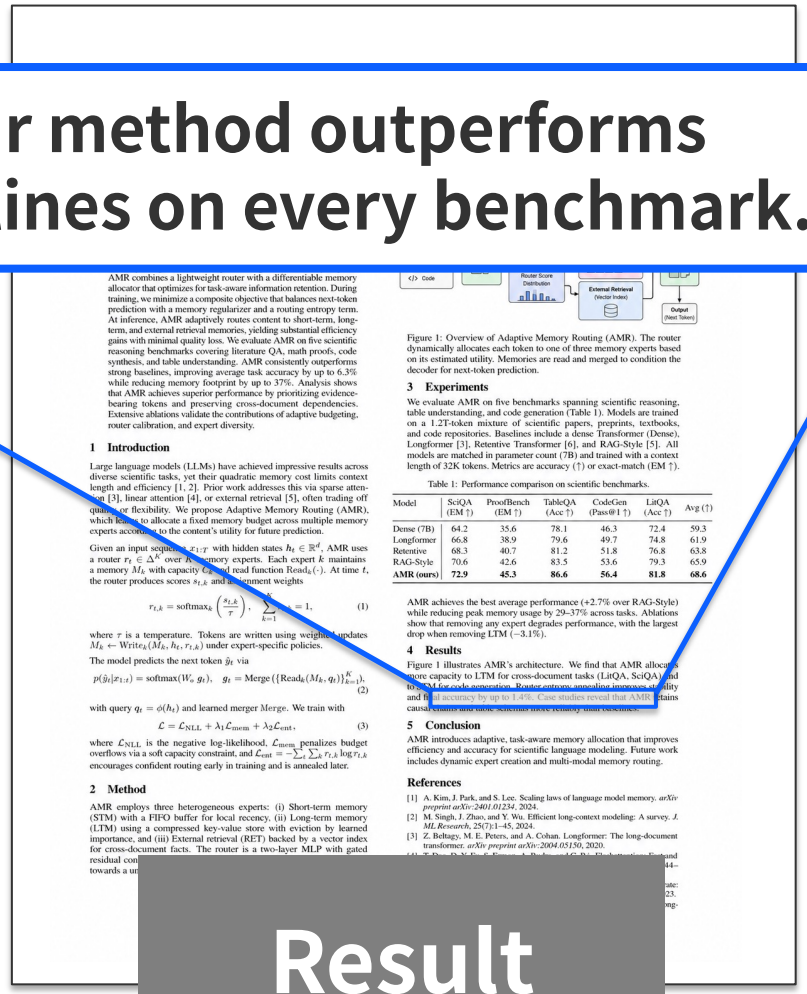
Results generated by AI research systems

Claim

"Our method outperforms all baselines on every benchmark."



Research System



Result

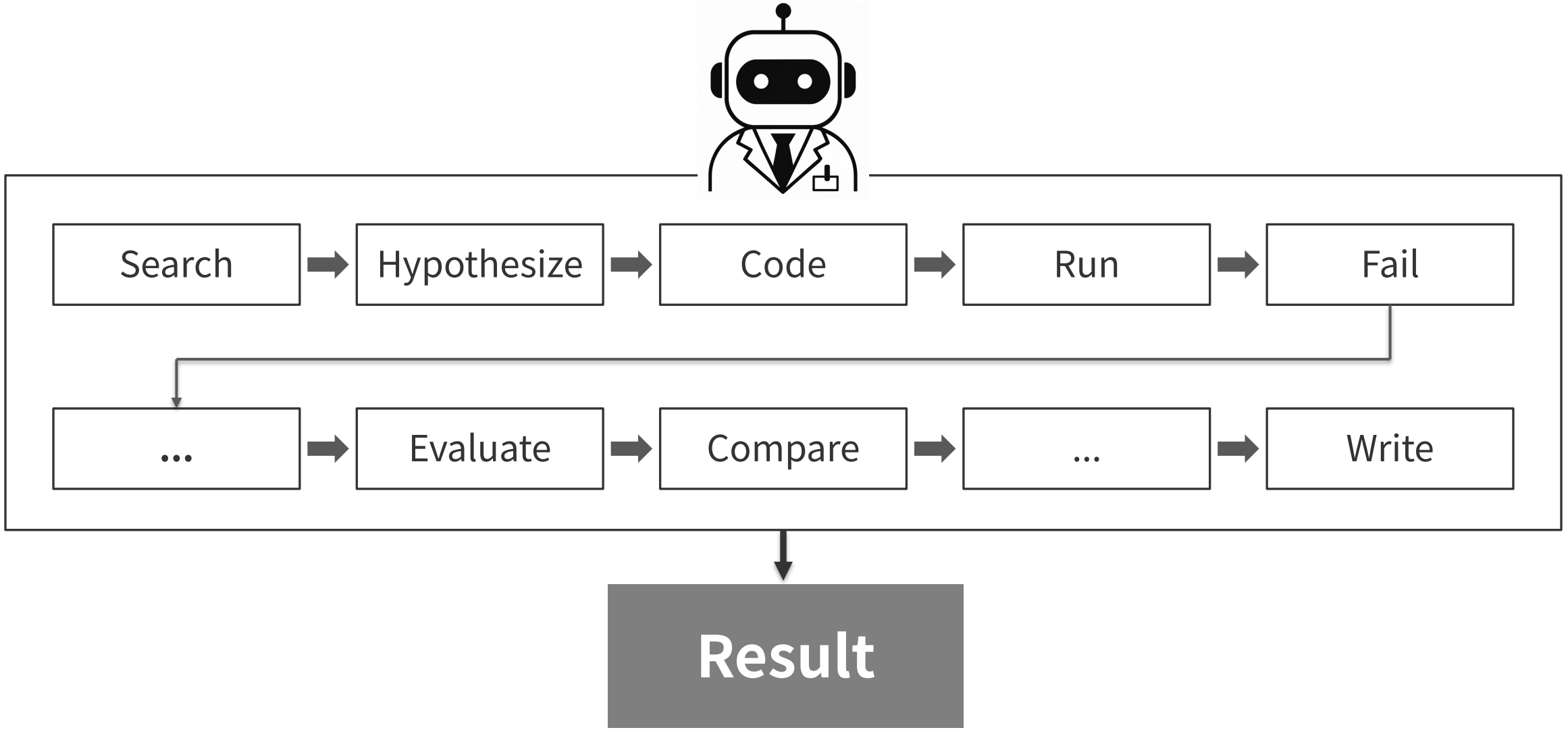
Reviewer

Was the comparison fair?

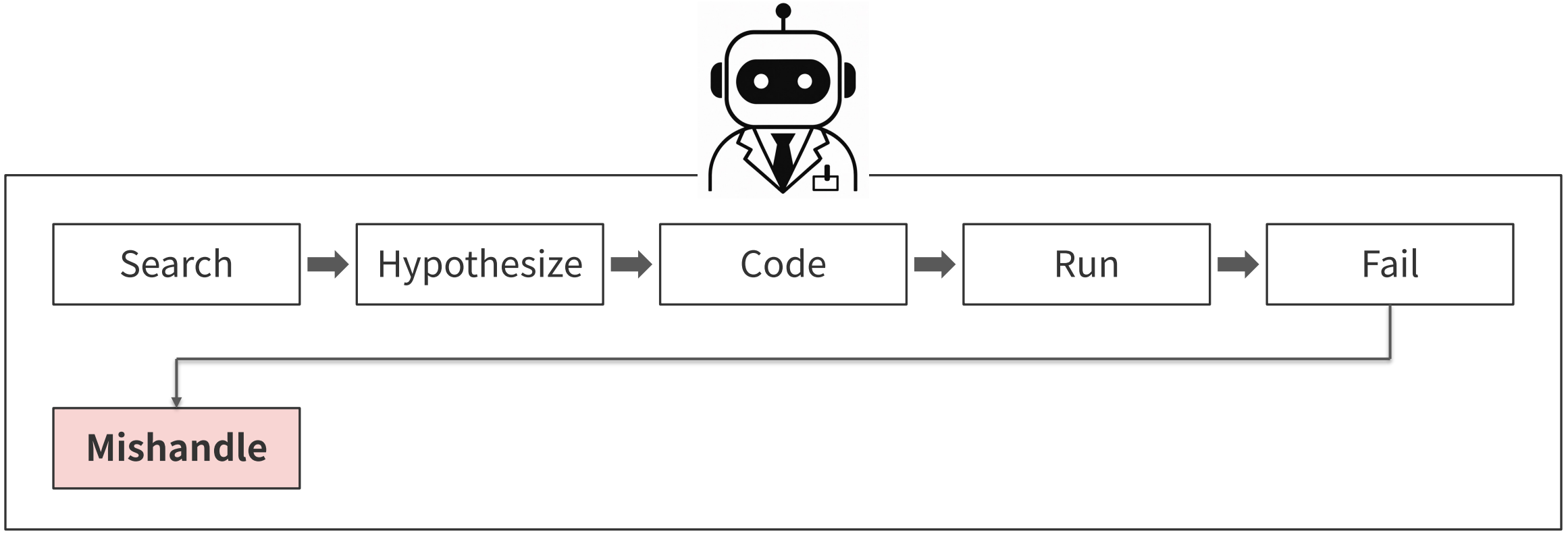
Are the results reproducible?

Did it show only the good results?

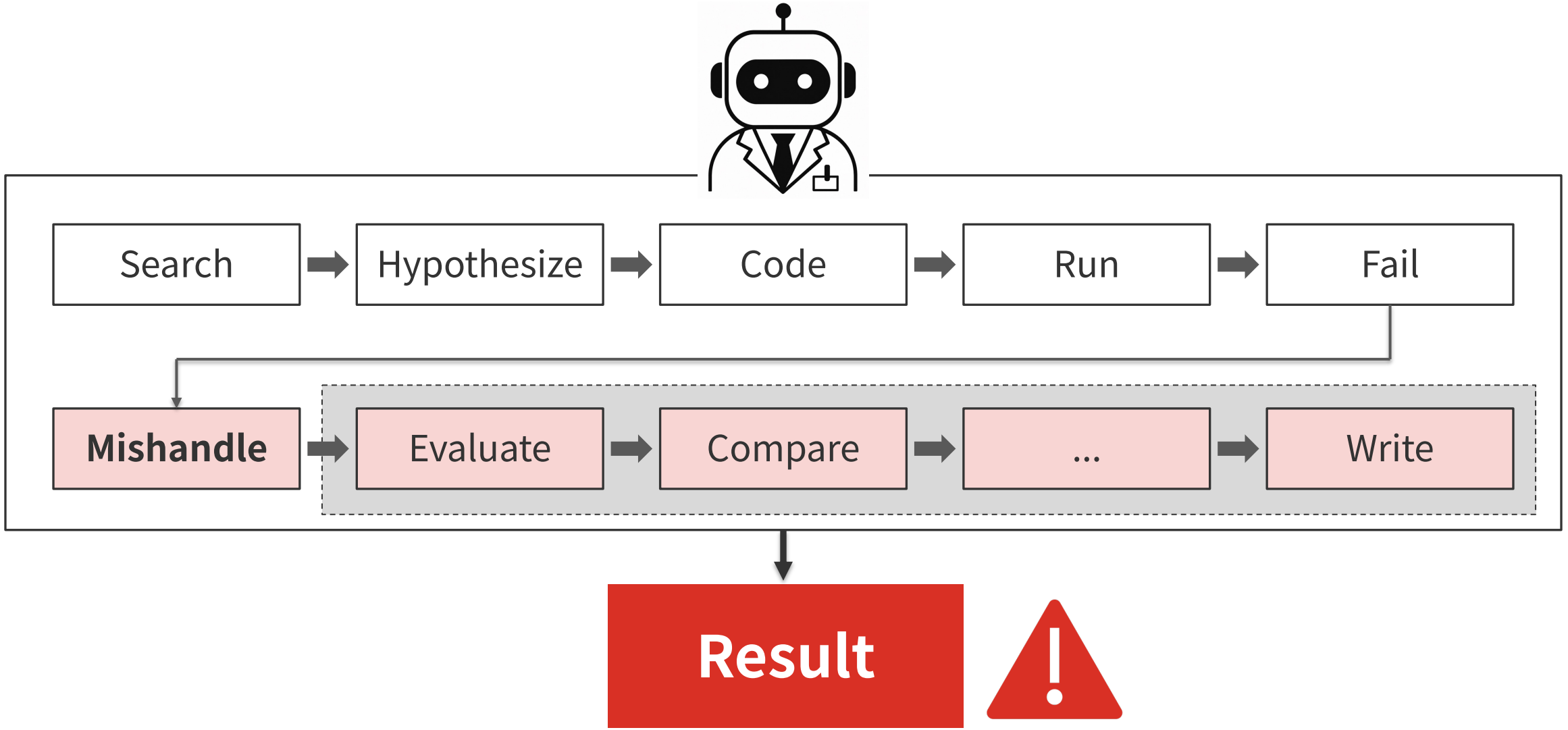
Inside AI research systems



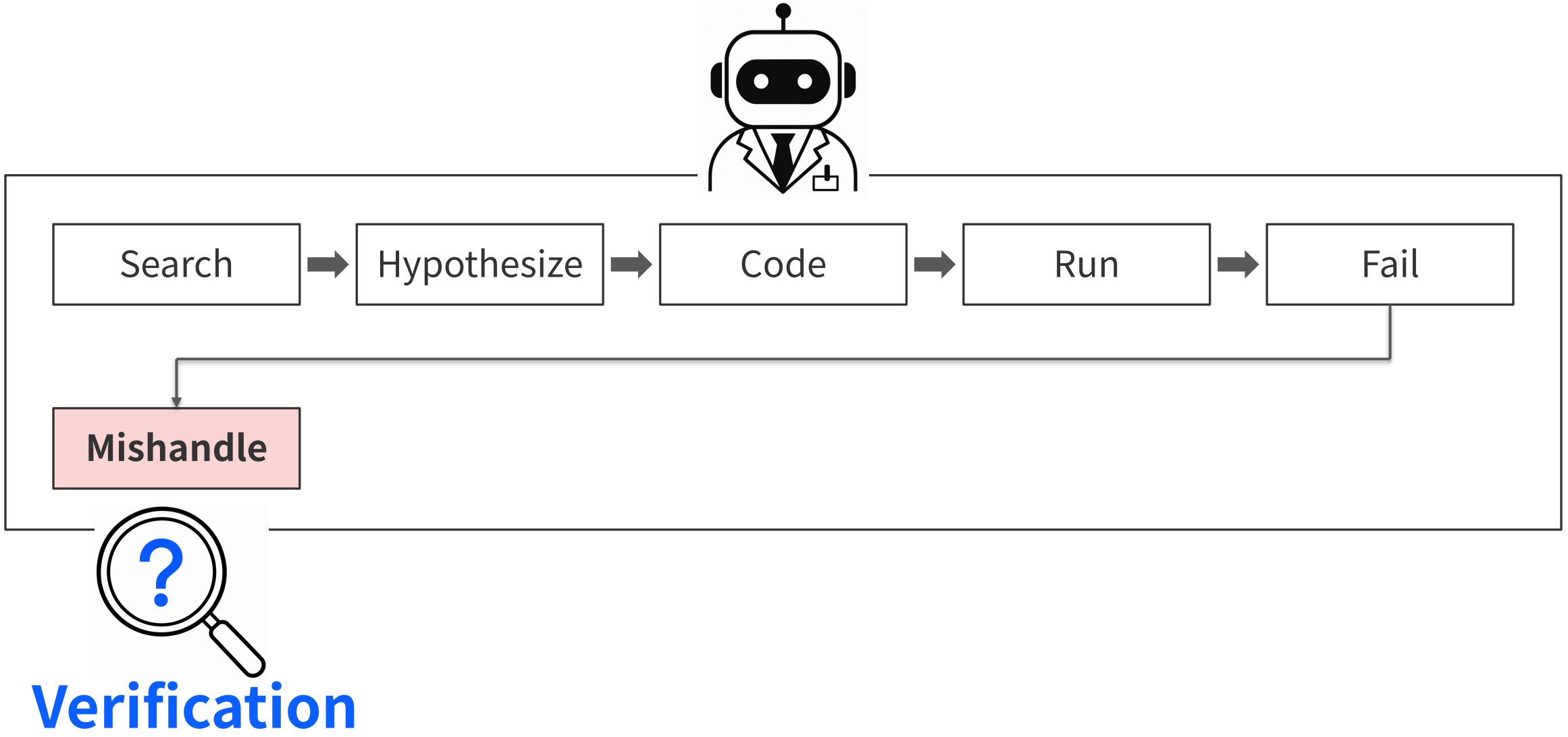
Inside AI research systems



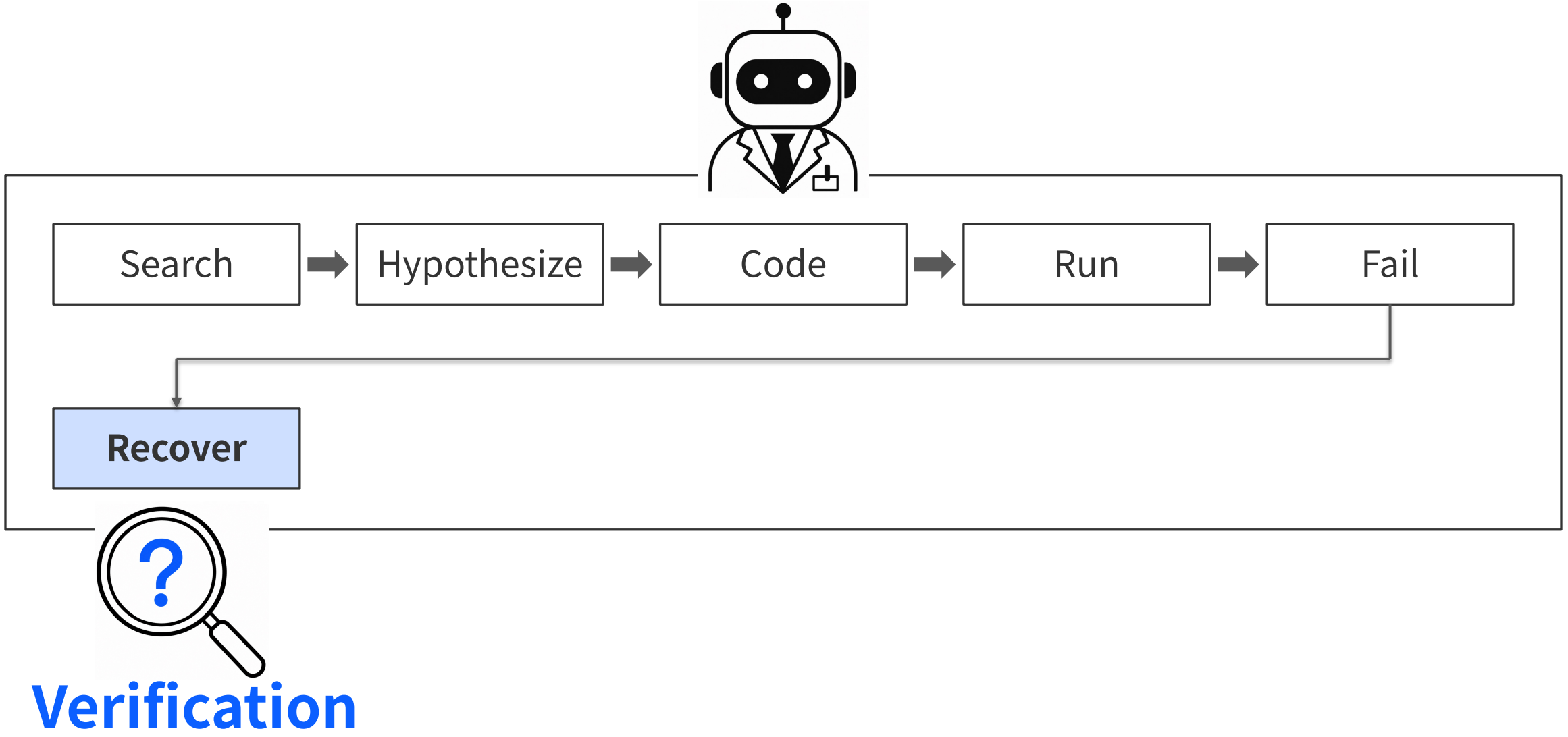
Inside AI research systems



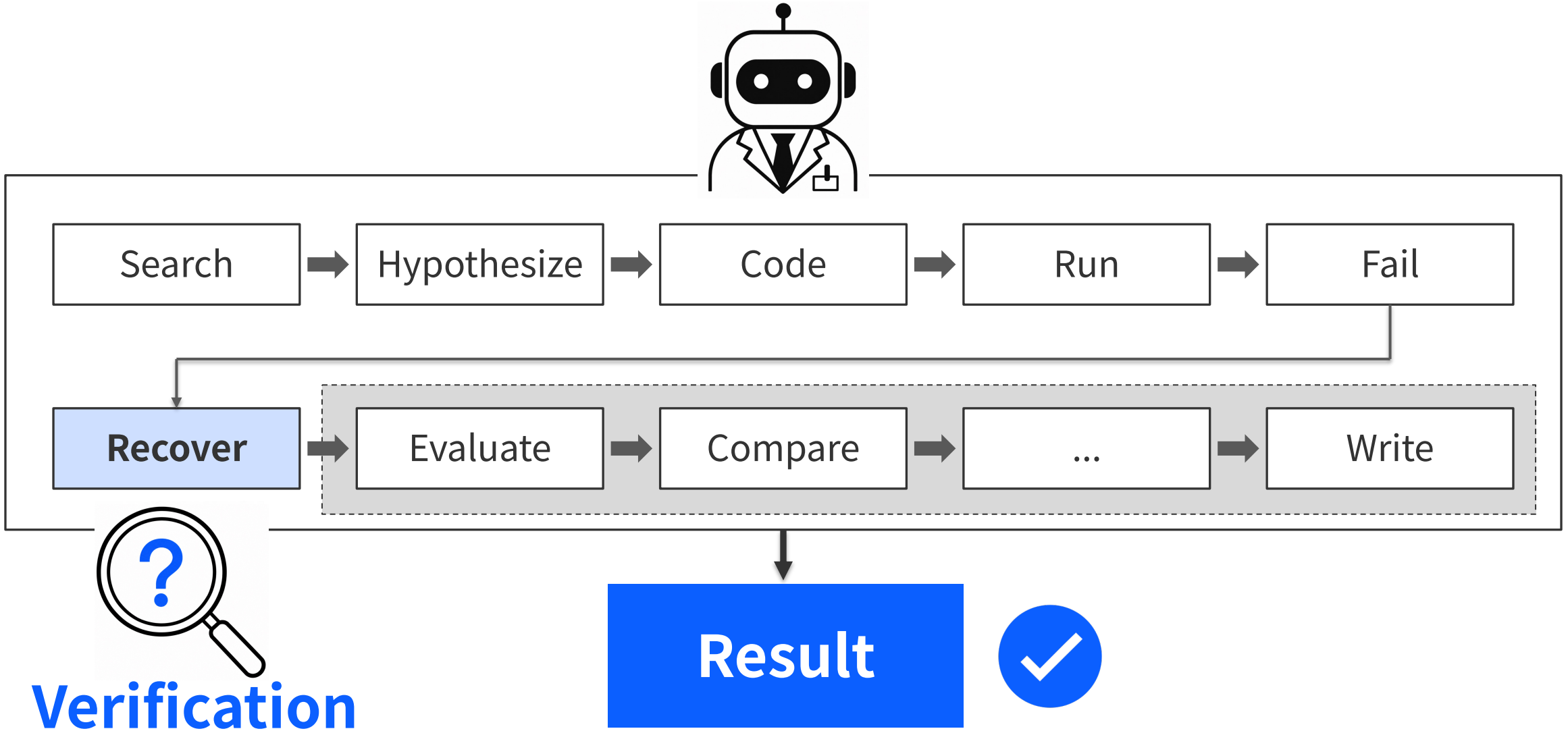
Inside AI research systems



Inside AI research systems



Inside AI research systems



To trust the results, we need verify the process.

Designing AI Research Systems for Verification

AIRAS

AI Research Automation System

Verifying claims throughout the research process

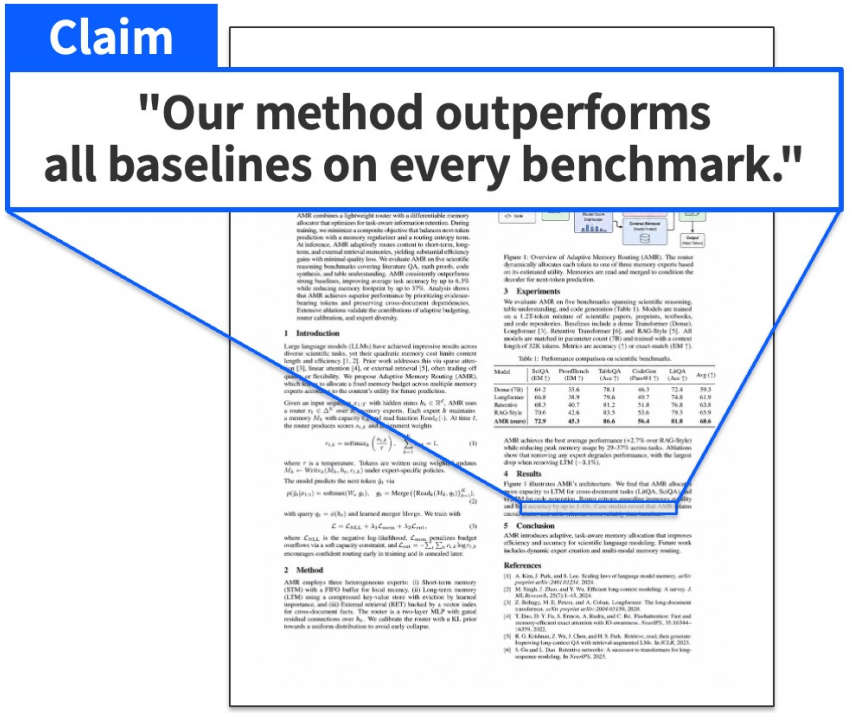
Goal

Trustworthy results from AI research systems

Approach

A research result consists of **claims** from each step of the process

→ **We make every claim verifiable**



Designing research for verification

Conventional research

Generate results

Decide how to verify

How to verify is decided **after** the research ends.

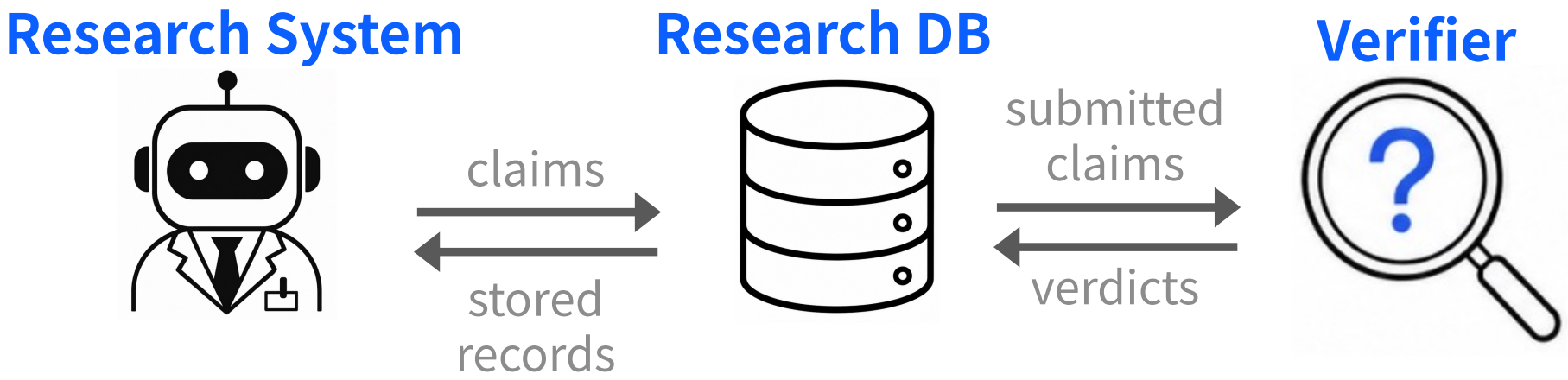
Verification-first research

Decide how to verify

Generate verifiable claims

How to verify is decided **before** the research starts.

Verification-first autonomous research



Three principles for verifiable research

1. Every claim is verifiable

Each claim says what holds and how to verify it

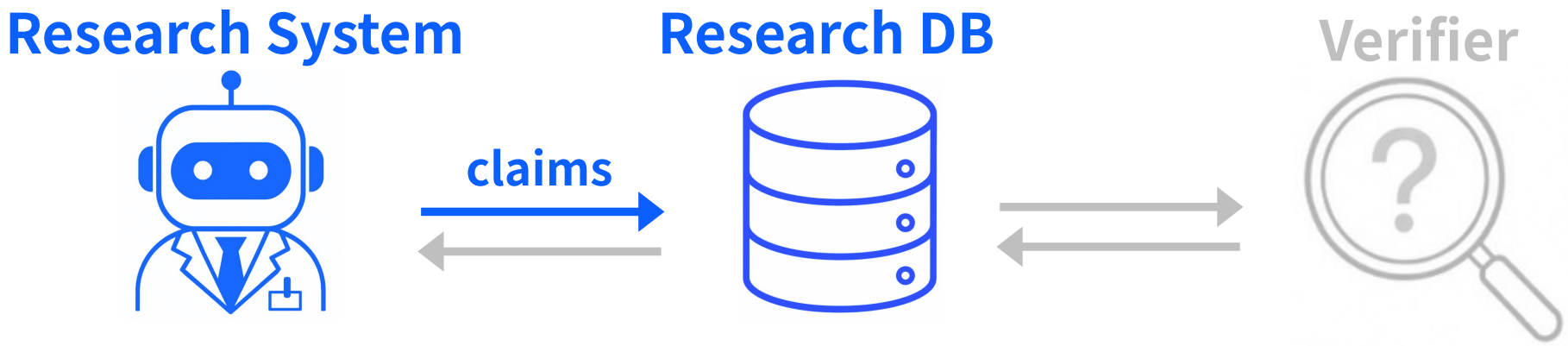
2. Every claim is verified outside the research system

The verifier is independent and the research system cannot influence it

3. Every claim has its source of truth in the research DB

Every paper is generated only from verified claims in the DB

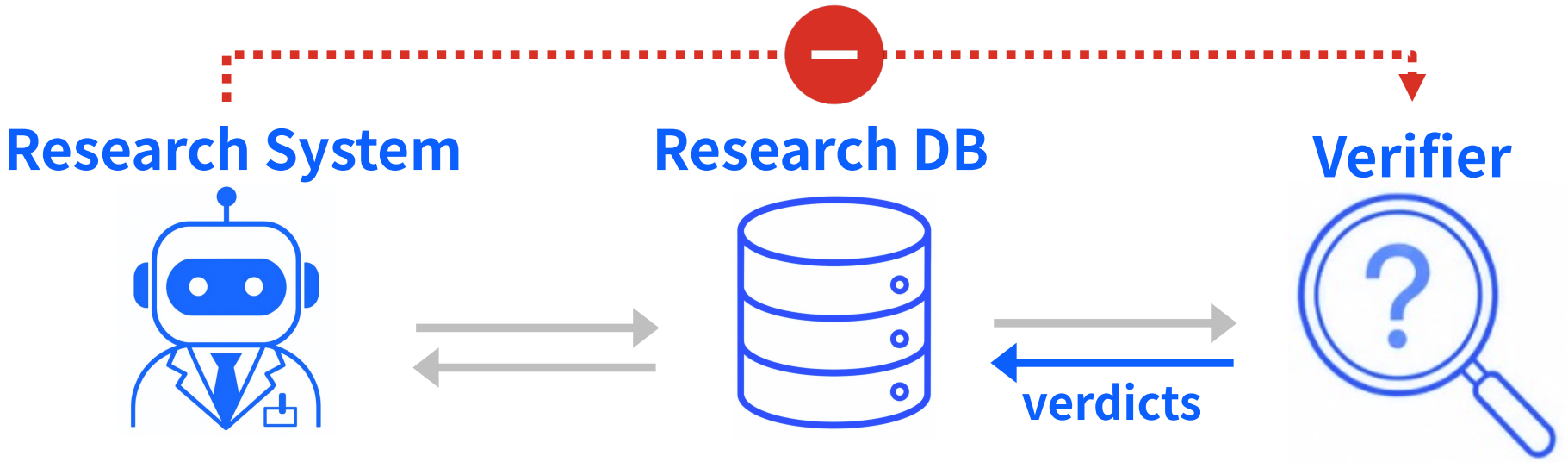
1. Every claim is verifiable




```
id: experiment-001
type: experiment
claim: "Method A outperforms Method B by 2.3 points on D."
code: "sha256:9f3c...a721"
check: rerun with new seeds
```

Each claim says what holds and how to verify it

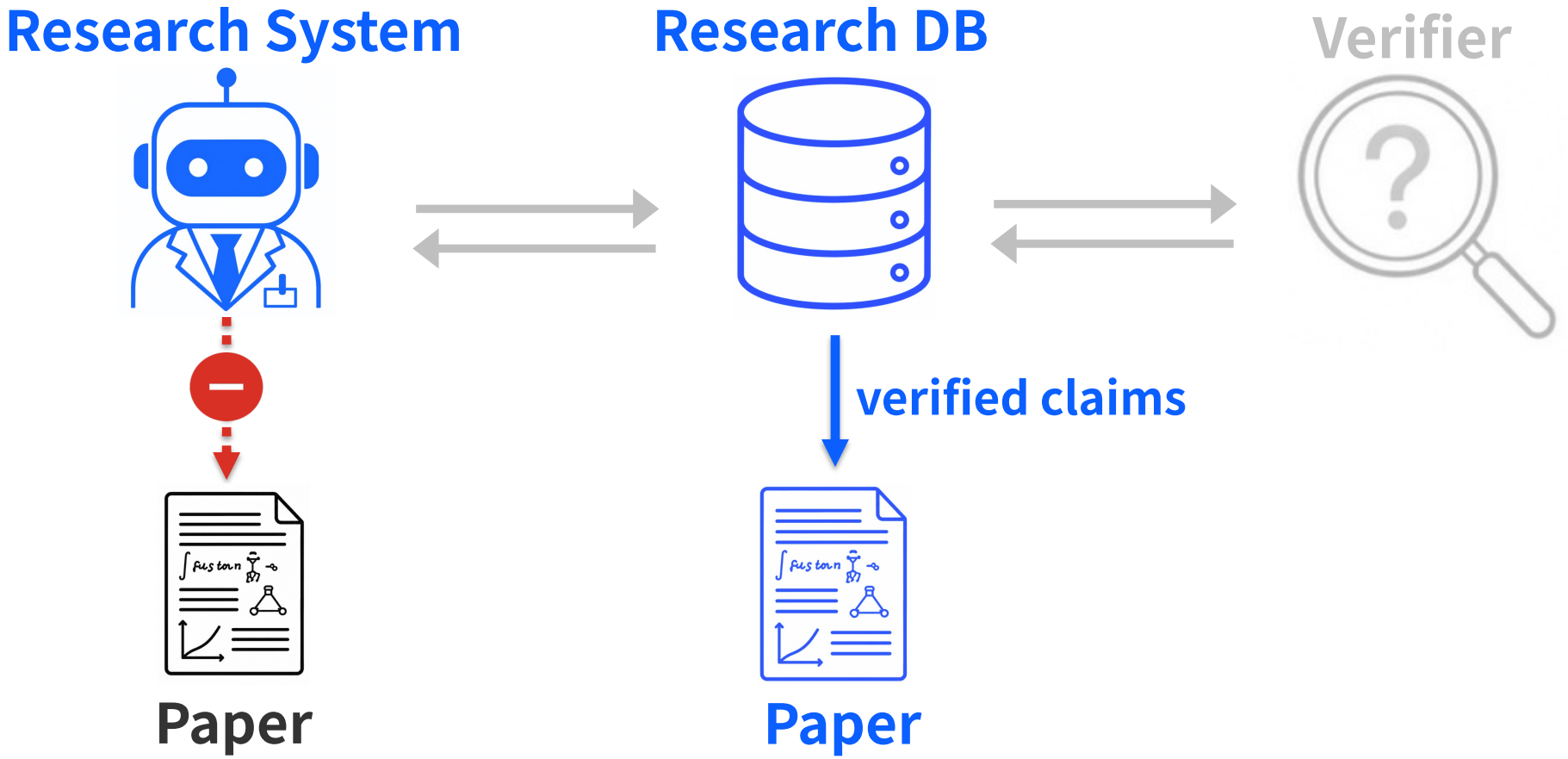
2. Every claim is verified outside the research system




```
claim_id: experiment-001
verifier: external-cluster-seyval
reported: +2.3
rerun: -1.1
verdict: FAIL
```

The verifier is independent and the research system cannot influence it

3. Every claim has its source of truth in the research DB



Every paper is generated only from verified claims in the DB

AIRAS: Key features

1. Open-source software (OSS) for research

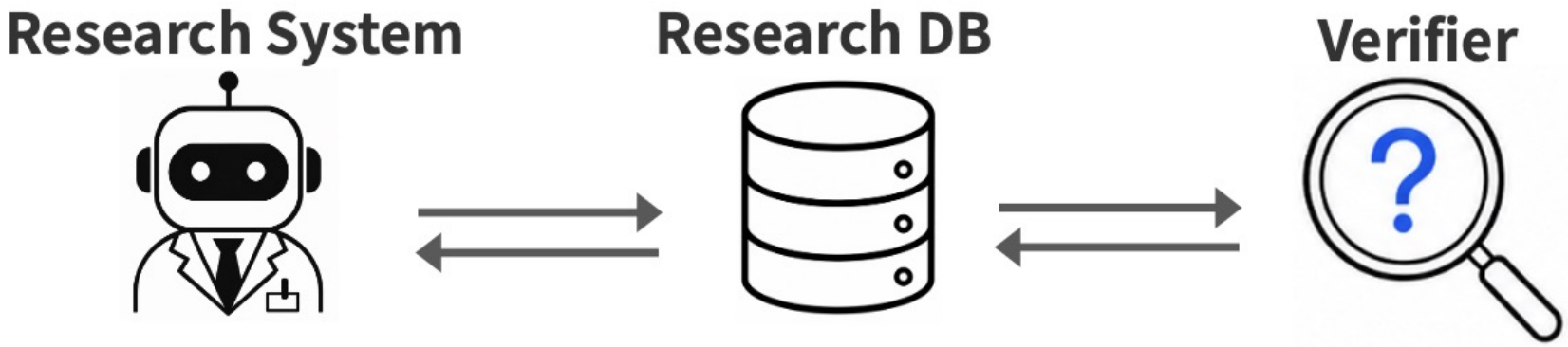
Anyone can use, modify, and extend AIRAS for their research.

2. Research tools via MCP

Different agents can access research tools through MCP.

3. Verification-first design

Designed to follow all three principles of verifiable research.



Conclusion

Conclusion

1. **AI** now makes discoveries and **speeds up research**
2. **Human verification can't keep up** with this speed
3. **AIRAS** automates verification for **trustworthy research**

AIRAS

Questions & Comments