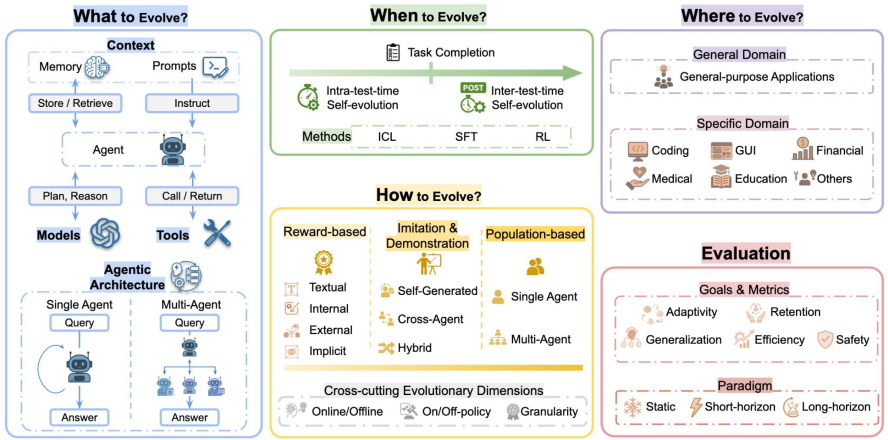
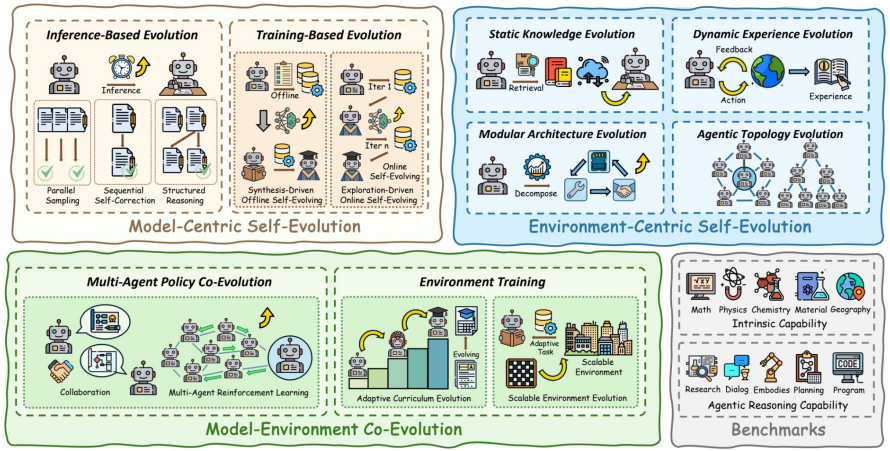
Position: Self-Play Only Evolves When Self-Synthetic Pipeline Ensures Learnable Information Gain

Revisit Self-Play and Self-Evolve From A Computational Complexity and Information-Theory Perspective

Wei Liu, Siya Qi, Yali Du, Yulan He

Self-Evolution





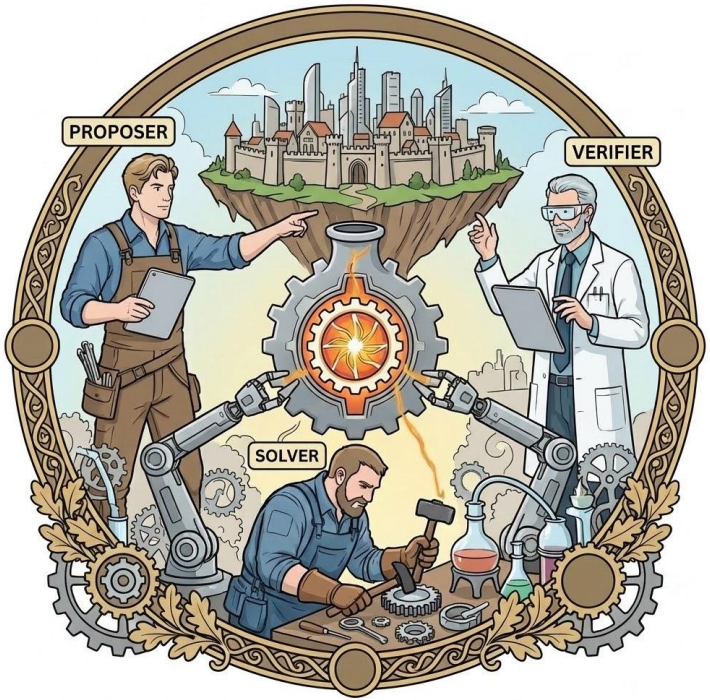
Various Definition:

- Continual Learning
- **Zero-data Self-Training**
- Prompting (update memory and experience)

A Survey of Self-Evolving Agents: On Path to Artificial Super Intelligence
A Comprehensive Survey of Self-Evolving AI Agents: A New Paradigm Bridging Foundation Models and Lifelong Agentic Systems
A Systematic Survey of Self-Evolving Agents: From Model-Centric to Environment-Driven Co-Evolution

Triadic Roles for Self-Evolution on Models

- **P**roposer, who propose the task
- **S**olver, who gives the solution for the task
- **V**erifier, who verify the solution

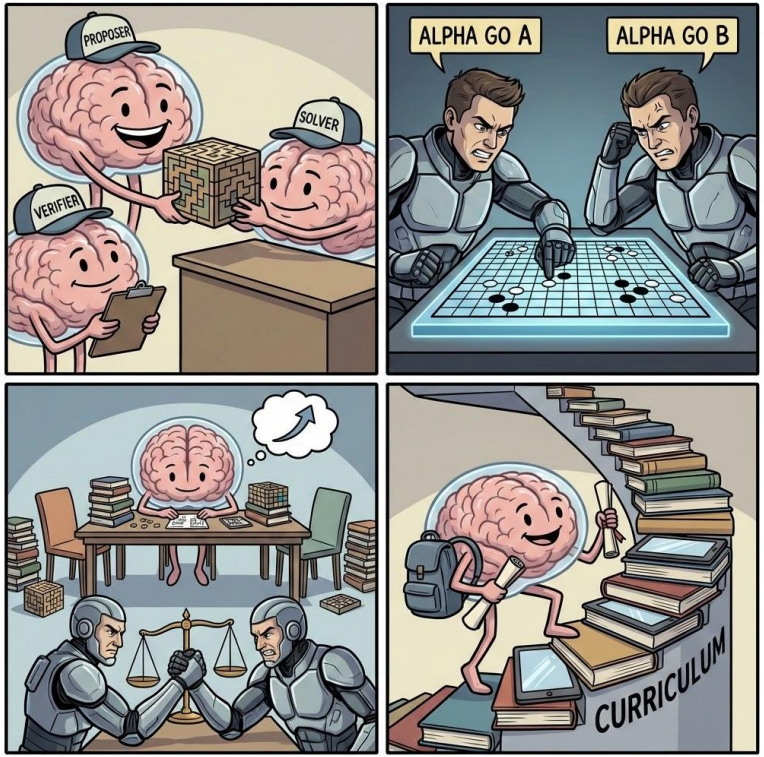


- Triadic for one (**P+S+V**) means all these three roles are put in one LLM
- So co(tri)-evolution becomes self-evolution

$$\theta^* = \arg\max_{\theta} \; V_\theta(P_\theta, S_\theta(P_\theta))$$

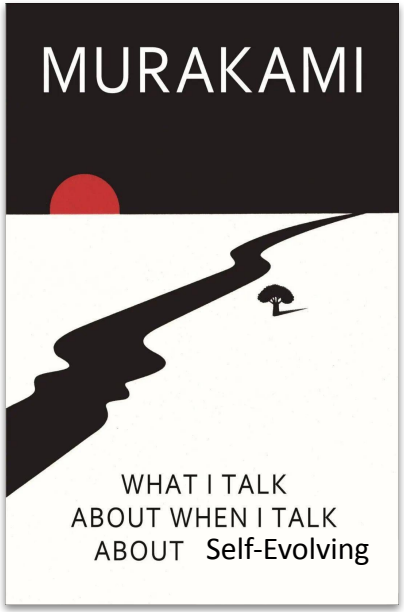
Not Real Self-Play

- Not zero-sum game
- No Nash Equilibrium
- More like autonomous Curriculum Learning
- So, what do we expect from such LLM self-play?



It’s self-synthetic data stream!

- Model *synthesis* its training data
- Model *synthesis* its feedback signal
- Model is trained on all these *synthetic* stuff



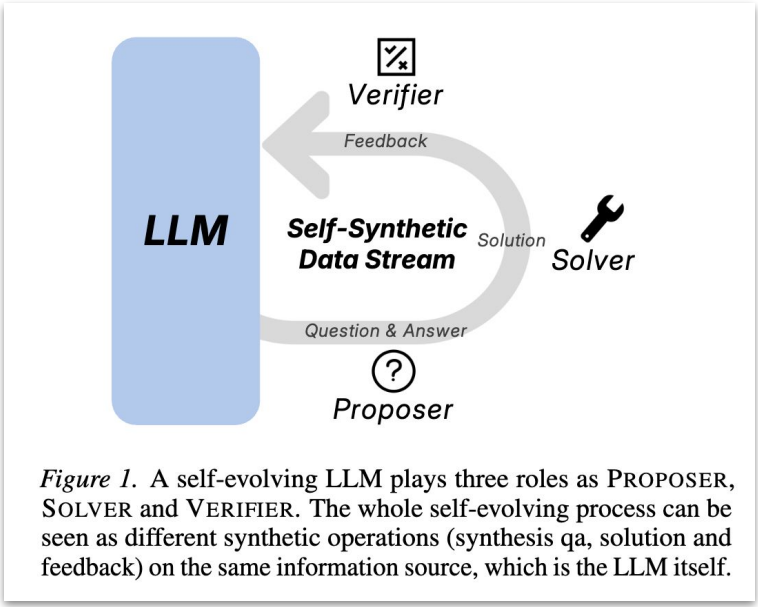
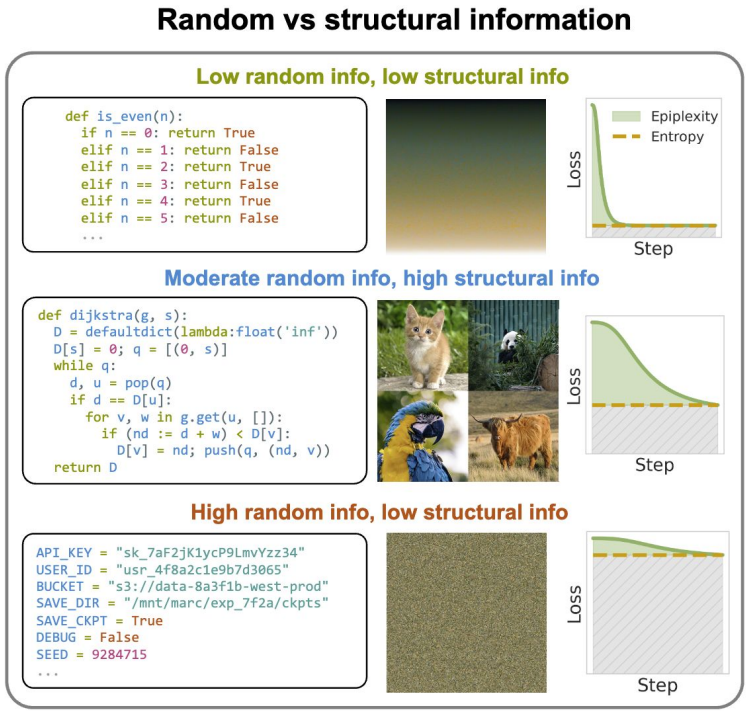


Figure 1. A self-evolving LLM plays three roles as PROPOSER, SOLVER and VERIFIER. The whole self-evolving process can be seen as different synthetic operations (synthesis qa, solution and feedback) on the same information source, which is the LLM itself.

Background on Epiplexity (Epistemic Complexity)



Definition 8 (Epiplexity and Time-Bounded Entropy) *Consider a random variable X on $\{0,1\}^n$. Let*

$$\mathrm{P}^\star = \arg\min_{\mathrm{P}\in\mathcal{P}_T} \{|\mathrm{P}| + \mathbb{E}[\log 1/P(X)]\} \tag{3}$$

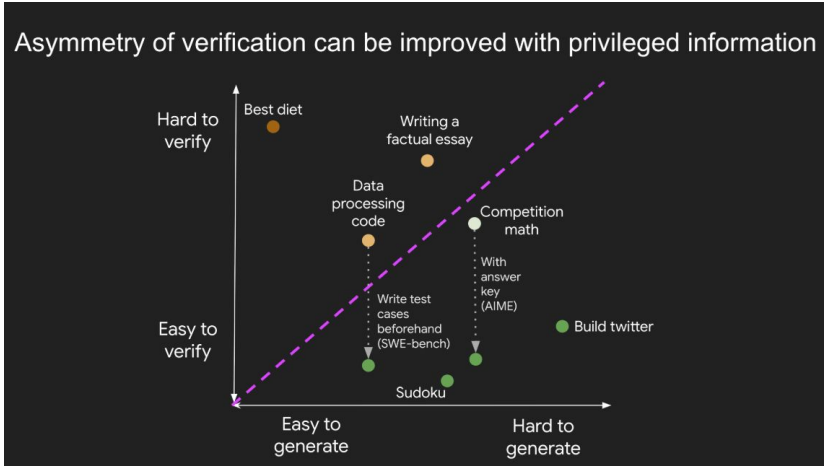
be the program that minimizes the time bounded MDL with ties broken by the smallest program, and expectations taken over X. $|\mathrm{P}|$ denotes the length of the program P in bits, and logarithms are in base 2. We define the T-bounded epiplexity S_T *and* entropy H_T *of the random variable X as*

$$\mathrm{S}_T(X) := |\mathrm{P}^\star|, \quad \textit{and} \quad \mathrm{H}_T(X) := \mathbb{E}[\log 1/P^\star(X)]. \tag{4}$$

It explains something…
And help us find what is missing…

1. Why Self-Play Learns: Asymmetry

- Explain:
  - **Computation creates learnable information → different synthetic direction yield different amount of learnable information**
  - So it creates information asymmetric gap
  - This gap motivates the weak-to-strong self-training



https://www.jasonwei.net/blog/asymmetry-of-verification-and-verifiers-law

1. Why Self-Play Learns: Asymmetry

- Explain:
  - **Computation creates learnable information → different synthetic direction yield different amount of learnable information**
  - So it creates information asymmetric gap
  - This gap motivates the weak-to-strong self-training

Figure 5. Epiplexity results on synthetic data with different tasks (induction, abduction and deduction) proposed by different PROPOSER LLMs and observed by different SOLVER LLMs. See details of calculating epiplexity in Appendix B.

1. Why Self-Play Learns: Asymmetry

- What is Missing?
  - Only weak verifier + proposer to strong solver, The ladder is not in closed loop
  - Only easy to verify → jagged intelligence
- Strong S → Strong V
  - Verifier-free RL
- Strong S → Strong P
  - Back-translation

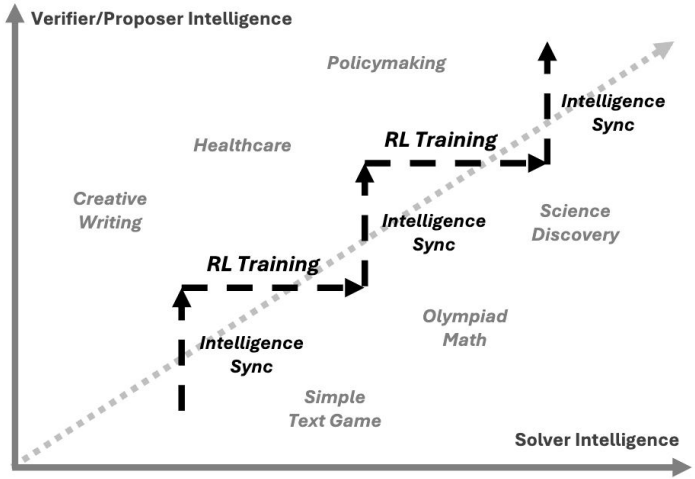


Figure 4. Climbing the intelligence asymmetry ladder by closing the loop among PROPOSER, SOLVER, and VERIFIER. "Intelligence synchronisation" denotes updating the weaker PROPOSER/VERIFIER with strong SOLVER. "Reinforcement learning" uses the weaker PROPOSER/VERIFIER to train the SOLVER.

Verifier-free RL: verifier co-evolve with solver

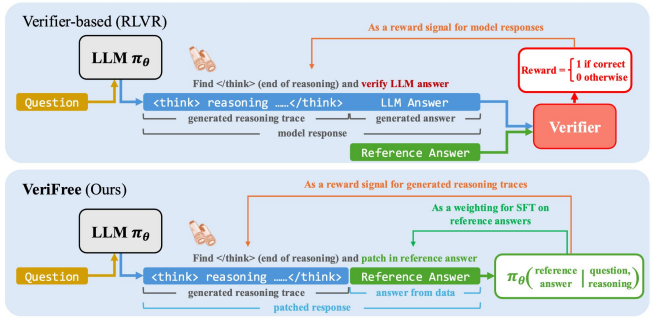


Figure 2: VeriFree enables R1-Zero-style LLM training without requiring access to a verifier. In the case of a single correct answer format, VeriFree optimizes exactly the same objective as R1-Zero with a lower variance gradient estimator.

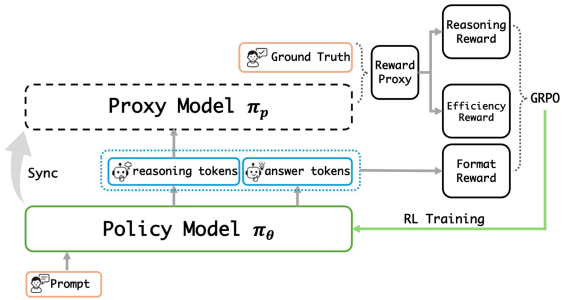
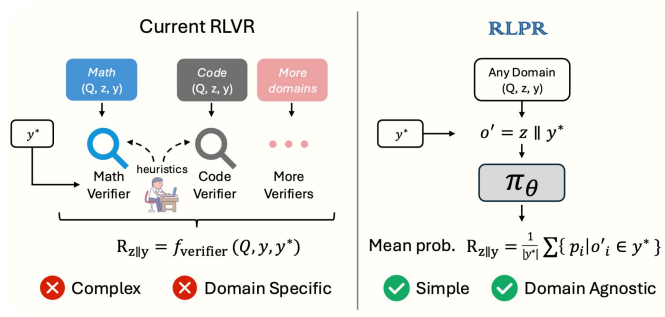


Figure 3: The overall process of NOVER.



- Three Verifier-free RL: VeriFree, NOVER, RLPR
- LLM acts as both the solver and verifier, **they share the same target**: maximize the probability to generate ground truth answer conditioned on its rollout reasoning parts

RLPR: Extrapolating RLVR to General Domains without Verifiers
Reinforcing General Reasoning without Verifiers
NOVER: Incentive Training for Language Models via Verifier-Free Reinforcement Learning

2. Why 50% acc for proposer?

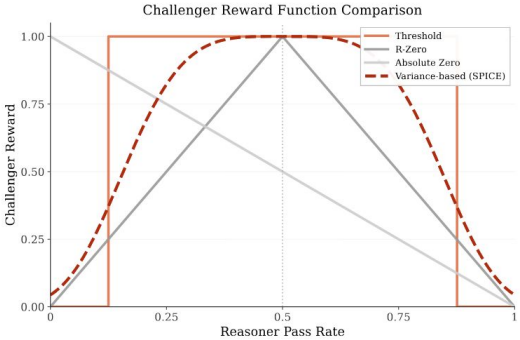


Figure 7 Comparison of challenger reward functions as a function of reasoner pass rate p. The **threshold reward** (rollout size 8) assigns 0 for trivial (8/8) and impossible (0/8) tasks, 1 otherwise. **Absolute Zero** $(1-p)$ linearly decreases with pass rate. **R-Zero** $(1-2|p-0.5|)$ peaks at maximum uncertainty. SPICE's **variance-based reward** (dashed) uses $\exp(-\frac{(\sigma^2-\sigma^2_{\text{opt}})^2}{2\cdot 0.01})$ where $\sigma^2 = p(1-p)$ and $\sigma^2_{\text{opt}} = 0.25$, creating a smooth curriculum centered at the frontier of model capability.

- Proposer raise questions to challenge solver with 50% pass ratio

2. Why 50% acc for proposer: Capacity Match

- Explain in **capacity-data match**
  - Too easy for model, no emergent learnable structure
  - Too hard for model, just unlearnable noise

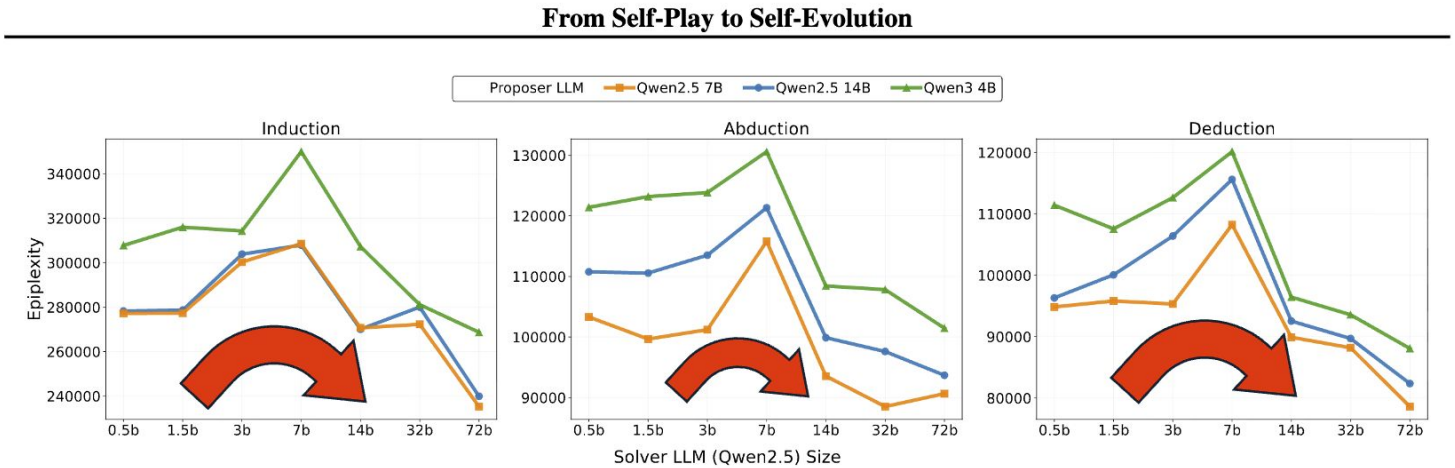


Figure 5. Epiplexity results on synthetic data with different tasks (induction, abduction and deduction) proposed by different PROPOSER LLMs and observed by different SOLVER LLMs. See details of calculating epiplexity in Appendix B.

2. What is Missing?

- Capacity Growth
  - **Information is f(data), Learnable information is f(data, model)**
  - Capacity of model should grow with self-evolving iteration
  - Capacity means spent computation, factors including model size, test-time budget, activated/useful component, etc

3. Shannon said: of course we need external information

- But in what fashion?
  - Give **proactivity** to model: proposer knows what to challenge solver
  - Seek for **asymmetry**: not just more data
  - Env-Model **Co-Evolution**: the environment shapes how information is presented to the model

What Three Designs Actually Shapes

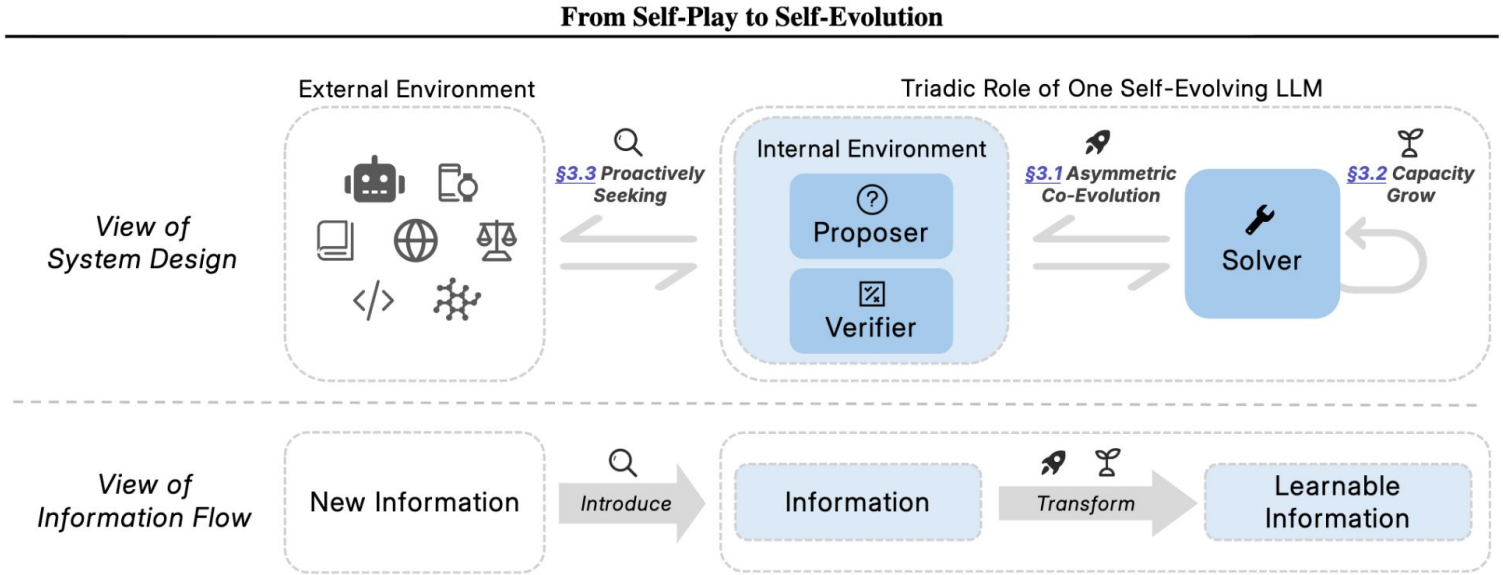


Figure 2. Overall framework of a triadic self-evolving loop. A self-evolving LLM plays three roles: the PROPOSER and VERIFIER form the internal environment, proactively interacting with the external environment to provide data and supervision for the SOLVER. The SOLVER and internal environment co-evolve asymmetrically, adaptively expanding capacity to capture more learnable information. From an information perspective, the system continually absorbs external information, and transform them into internal learnable information.

What Three Designs Actually Shapes

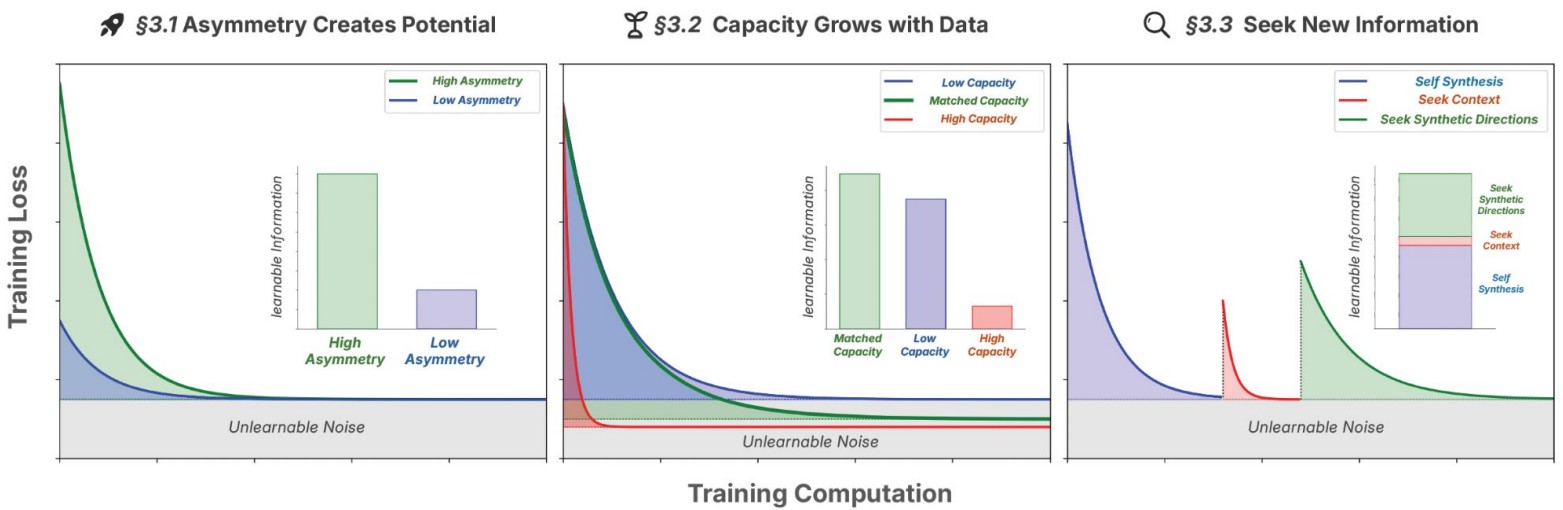


Figure 3. Illustration of three designs from the perspective of learnable information. Asymmetry between the SOLVER and the PROPOSER/VERIFIER creates learning opportunities. Expanding model capacity to match self evolving data opens space for learnable information. Reusing the same patterns in new contexts yields limited gains, whereas introducing new synthetic directions creates fresh asymmetries and thus new sources of learnable information.

Learnable Information Gain in Self-Play is Unstable

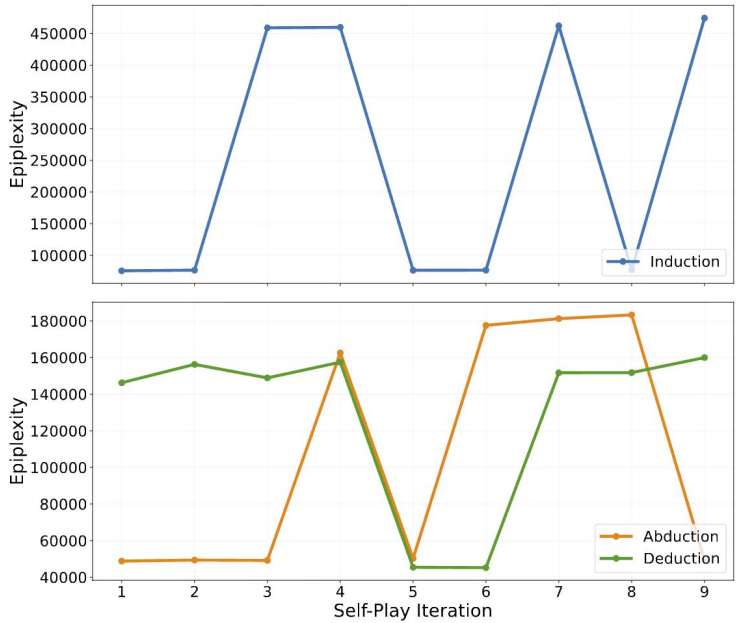


Figure 6. Epiplexity results during the self-play training on three tasks. See task detail in Appendix C.

- Across the three coding tasks (Induction, Abduction and Deduction), the learnable information in self synthesized data varies substantially across self play iterations.

Call to Action

- Sustainable Self-Evolving
  - three missing designs based on learnable information
- Beyond the Math&Code
  - cross the easy-to-verify side, llm application in the wild
- Beyond the RL
  - is critic-free, model-free, policy-based RL all you need?
  - is countless-hyperparams learning algorithm all you need?
- Monitoring and Evaluation
  - from chaotic RL into auditable data engineering
  - clean and robust evaluation