

The Asymmetry Engine, Chapter 1: Intelligence is a map

Martin Roark

Chapter 1 of *The Asymmetry Engine: How AI Closes the Gaps, and What Stays Scarce*, October 2026. The online edition, with the interactive figures and the links between chapters, is at <https://future-seems-so-good.com/blog/the-asymmetry-engine#map>. License: CC BY 4.0.

1:1 A task is a description and the set of answers that would solve it. Intelligence is the best map from descriptions to answers that a compute budget can buy, scored by verified success, so its measure is a curve: the best verified success at each budget. The definition puts a verifier inside intelligence, and the verifier is the root object of this book. A sound verifier turns compute into a training signal, and an unsound one turns it into Goodhart's law. Three costs (to check an answer, to construct a solved example, to find a solution) decide which maps can be harvested cheaply.

1.1 The function that maps

1:2 A model that answers prompts is a function: task descriptions go in, candidate solutions come out. Scoring the function takes a third object, a procedure that decides whether an answer counts.

1:3

Definition 1.1 (Intelligence as a map). Tasks $x \in \mathcal{X}$ are drawn from a task distribution $\mathcal{D}$, candidate solutions are $y \in \mathcal{Y}$, and $\mathcal{Y}^*(x) \subseteq \mathcal{Y}$ is the set of correct solutions of x . A **verifier** $V : \mathcal{X} \times \mathcal{Y} \rightarrow \{0, 1\}$ accepts a candidate ($V = 1$) or rejects it. A policy f maps each task to a candidate $f(x)$ and spends expected compute $C(f)$ per task; if it maps at random, expectations also run over its draws. For a compute budget B , the best policy within the budget is f_B^* , and its score $\mathcal{I}(B)$, read as a function of the budget, is the **intelligence curve** of (1.1). One policy on one task family is a skill; intelligence is the curve.

1:4

$$f_B^* = \arg \max_{f: C(f) \leq B} \mathbb{E}_{x \sim \mathcal{D}} [V(x, f(x))], \quad \mathcal{I}(B) = \max_{f: C(f) \leq B} \mathbb{E}_{x \sim \mathcal{D}} [V(x, f(x))] \quad (1.1)$$

1:5 Three features of (1.1) carry most of the later argument. The first is that it is a curve. $\mathcal{I}(B)$ never falls as B grows, because every policy that fits a budget fits any larger one. A more intelligent system has a higher curve, and a cheaper model that reaches the same score at a lower budget has raised the curve at the budgets where most tasks are bought (§3.1).

1:6 The curve is steep even on a benchmark built to resist bought skill. In December 2024, OpenAI's o3 scored 75.7% on the semi-private set of ARC-AGI-1 within the public leaderboard's \$10,000 compute limit, and 87.5% in a configuration that used about 172 times as much compute ([ARC Prize](#), "[OpenAI o3 Breakthrough High Score on ARC-AGI-Pub](#)"). One model had two scores, one per budget.

1:7 The second feature is that the curve is relative to a distribution. Change $\mathcal{D}$ and two policies can swap places: a model that is strong at olympiad geometry can be weak at tax law. The third is that it is relative to a verifier. The objective contains V , and the correct answers $\mathcal{Y}^*$ enter only through it, so a learner can optimize only what can be checked. Where V is cheap and faithful, compute turns into competence. Where it is expensive, noisy or open to reward hacking, more compute buys little.

1:8 Without the budget, (1.1) is the objective of reinforcement learning with the verifier as reward. A one-step problem observes $x \sim \mathcal{D}$, emits y and is paid $V(x, y)$; a multi-step agent replaces y with a trajectory and V with a check on its outcome. Jason Wei put the equivalence in one line: "In RL terms, ability to verify solutions is equivalent to ability to create an RL environment" ([Wei](#), "[Asymmetry of verification and verifier's law](#)"). It is the training objective of the reasoning models of 2025, whose record is in §2.6.

1:9 Training against a verifier, with a penalty that keeps the policy near its base, has a closed-form optimum: the base policy's probabilities, tilted toward the answers the verifier accepts (Proposition 2.3). The tilt cannot create an answer the base policy never produces, and it favors whatever the verifier accepts, correct or not (Proposition 1.1). A trained map is a prior reshaped by a verifier, and both have to be good.

1.2 Where the definition sits

1:10 The map from tasks to answers is the standard formalism for a task, and the standard definitions of intelligence sit one level above it. Shane Legg and Marcus Hutter's informal version is "Intelligence measures an agent's ability to achieve goals in a wide range of environments." Their formal version sums a policy's performance over every computable environment and weights each environment by its simplicity (Legg & Hutter, "Universal Intelligence"):

$$\mathcal{I}_{\text{LH}}(f) = \sum_{\text{env}} 2^{-|\text{env}|} \text{reward}_{\text{env}}(f)$$

Here $|\text{env}|$ is the length in bits of the shortest program that generates the environment, its Kolmogorov complexity, and $\text{reward}_{\text{env}}(f)$ is the expected total reward the policy earns there. Ashby's reading of intelligence as appropriate selection reaches the same level from cybernetics (§4.2).

1:11 (1.1) restricts Legg and Hutter's sum in three ways: a task distribution someone chose replaces the weighting by simplicity, a verifier replaces reward, and a compute budget is added. $\mathcal{I}_{\text{LH}}$ is incomputable, because Kolmogorov complexity is; $\mathcal{I}(\mathcal{B})$ can be measured. The price of measurability is parochialism. The curve measures ability on the tasks someone thought to verify, which makes the choice of verifiers a strategic act. Generality is the curve averaged over task families with weights set by economic value, a priced version of the same sum, because an economy pays for solving valuable problems and Legg and Hutter's weights favor simple ones.

1:12 François Chollet puts intelligence higher still: "The intelligence of a system is a measure of its skill-acquisition efficiency over a scope of tasks, with respect to priors, experience, and generalization difficulty." His reason is that "unlimited priors or unlimited training data allow experimenters to 'buy' arbitrary levels of skills for a system, in a way that masks the system's own generalization power" (Chollet, "On the Measure of Intelligence"). (1.1) measures a skill at a moment. Chollet's quantity measures how fast a learner raises its curve per unit of experience on tasks unlike the ones it trained on. Harvesting solved examples raises the first (§2.3), and recursive self-improvement is a claim about the second (§11.1).

1:13 Skill can be bought on Chollet's own benchmark. The MIT and Cornell entry that reached 47.5% on ARC-AGI-1's semi-private set in 2024 "improved after training on 400k synthetically-generated ARC tasks" (ARC Prize, "2024 Progress on ARC-AGI-Pub"). Synthetic tasks raised skill on the benchmark's distribution. Whether such skill transfers to tasks unlike them is the question Chollet's definition asks, and (2.2) bounds the answer.

1:14 Leonid Levin's universal search is the limit of finding by search. To find y with $V(x, y) = 1$, run every program f in parallel, giving each a share $2^{-|f|}$ of the time, where $|f|$ is its length in bits. If some program solves x in time $t_f(x)$, the search succeeds within

$$T_{\text{Levin}}(x) \leq 2^{|f|+1} t_f(x)$$

(Levin 1973; Scholarpedia, "Universal search"). The search is optimal up to the factor $2^{|f|}$, the price of not knowing which program to run. A trained policy is a prior that puts its mass on good programs, so training shrinks the factor, and knowledge replaces enumeration. The search consults a verifier at every step: without a verifier there is no universal search.

1:15 The fourth classical view equates prediction with compression. By arithmetic coding, a model that gives the next symbol probability p becomes a lossless code that spends $-\log_2 p$ bits on it, and any such code is a model. Grégoire Delétang and colleagues showed how far this reaches: Chinchilla 70B, trained mainly on text, compresses ImageNet patches to 43.4% of their raw size and LibriSpeech samples to 16.4%, beating PNG (58.5%) and FLAC (30.3%) (Delétang et al., "Language Modeling Is

Compression"). In the terms of (1.1), pretraining is the case in which the verifier is free. The corpus supplies the task (the context) and the answer (the next token), and the check is a lookup. That verifier costs nothing, and its supply is finite (§2.7).

1.3 Sound verifiers

1:16 A verifier can err in two directions, and training tolerates only one of them.

1:17

Definition 1.2 (Soundness and completeness). **Soundness** means $V(x, y) = 1 \Rightarrow y \in \mathcal{Y}^*(x)$: the verifier never accepts a wrong answer. Completeness means $y \in \mathcal{Y}^*(x) \Rightarrow V(x, y) = 1$: it never rejects a right one. A verifier with these properties is *sound* and *complete*. The true success of a policy is $U(f) = \Pr_{x \sim \mathcal{D}} [f(x) \in \mathcal{Y}^*(x)]$, and its verified success is $\hat{U}(f) = \mathbb{E}_{x \sim \mathcal{D}} [V(x, f(x))]$.

1:18 Splitting the event $V = 1$ by whether the answer is correct gives an exact identity:

$$\hat{U}(f) - U(f) = \epsilon_+(f) - \epsilon_-(f), \quad \epsilon_+ = \Pr [V = 1, f(x) \notin \mathcal{Y}^*(x)], \quad \epsilon_- = \Pr [V = 0, f(x) \in \mathcal{Y}^*(x)]$$

1:19 Here ϵ_+ is the verifier's false-accept mass and ϵ_- its false-reject mass. A sound verifier has $\epsilon_+ = 0$, so it can only under-credit: $\hat{U} \leq U$ for every policy. An incomplete verifier wastes signal, and an unsound one can be inflated.

1:20

Proposition 1.1 (Soundness is what a training signal must keep). (i) If V is sound, every rise in $\hat{U}$ certifies a rise in a lower bound on U . (ii) If V is unsound and the policy class can reach its false accepts, a maximizer of $\hat{U}$ need not maximize U , and an optimizer held to a budget may prefer reaching false accepts to solving. (iii) If V is complete, the share of accepted answers that are correct is

$$\frac{U}{\hat{U}} = \frac{U}{U + \epsilon_+},$$

which falls below one half once $\epsilon_+ > U$. For a verifier that accepts each wrong answer with a fixed probability, this happens roughly once the policy's success rate falls below that probability. There most rewarded behavior is the hack, and training reinforces it.

▼ Proof

(i) is (1.2) with $\epsilon_+ = 0$, which gives $U \geq \hat{U}$ for every policy. For (ii), take any policy f and a policy f' that agrees with it except on a share s_{fa} of tasks where f answers wrongly and is rejected; there f' emits a wrong answer that V accepts. Then $\hat{U}(f') = \hat{U}(f) + s_{fa}$ while $U(f') = U(f)$. If reaching those false accepts costs less compute than solving, the budget in (1.1) can make f' preferable to a policy with higher U , and an optimizer that sees only V cannot tell them apart. For (iii), completeness gives $\epsilon_- = 0$, so every correct answer is accepted and $\hat{U} = U + \epsilon_+$. If V accepts each wrong answer with the same probability, ϵ_+ is that probability times $1 - U$, and $\epsilon_+ > U$ exactly when U is below that probability divided by one plus it. ■

1:21 Proposition 1.1 is Goodhart's law stated for verifiers. In Marilyn Strathern's wording, "When a measure becomes a target, it ceases to be a good measure" ([Strathern, "Improving ratings: audit in the British University system", 1997](#)). An optimizer sees only the verifier, so the pressure of training flows to wherever the verifier and the truth disagree, and the measured record of that pressure is in §2.6. The builders of training environments interviewed for Epoch AI's January 2026 survey ranked reward hacking as their first quality concern, and one put the proposition in a sentence: "Soundness matters most: high reward must mean the task was actually solved, not hacked" ([Epoch AI, "An FAQ on Reinforcement Learning Environments"](#)).

1:22 The soundness a verifier needs grows with the intelligence it trains. A verifier that checks the defining rules of a task, such as the Sudoku constraints or a proof assistant's kernel, has a false-accept set that is empty or small enough to audit (§2.2). A verifier that compares against a stored answer, or is itself a learned model, has false accepts, and a more capable optimizer reaches more of them. So verification and alignment become one problem at the frontier: a value that cannot be checked cannot be optimized directly, and a verifier open to reward hacking will be hacked by a strong enough optimizer (§13.3; §13.4).

1.4 The cost triangle

1:23 Whether a task family is cheap to learn is set by three costs and how they compare.

1:24

Definition 1.3 (The cost triangle). For a task family with verifier V , measure three costs in one unit (elementary operations, FLOPs or dollars): c_{ver} to check a given candidate, c_{con} to construct a solved pair (x, y) with $y \in \mathcal{Y}^*(x)$, usually backward from y , and c_{sol} to find an accepted y from x alone. Reinforcement learning harvests the ratio $c_{\text{sol}}/c_{\text{ver}}$, the verification asymmetry of Definition 2.1; supervised learning harvests $c_{\text{sol}}/(c_{\text{con}} + c_{\text{ver}})$, as in (2.1).

1:25 Each ratio is an asymmetry, finding against checking in one case and finding against building and checking in the other, and the two ratios name two harvests. A reward is a verifier, so reinforcement learning needs only a cheap check. This is the picture of the class NP, where checking a certificate takes polynomial time and finding one may take exponential time. A labeled example needs the answer itself, so supervised learning needs a cheap construction: draw the answer first, then derive its problem. Some families have neither. In taste, strategy, research judgment and values, checking an answer costs about as much as producing it, or no verifier is well defined, and these families resist both harvests (Proposition 2.2; §13.3).

1:26 The three costs bound one another through the solver. A solver that samples until the verifier accepts pays for at least one check per try, and its cost of finding falls as its pass rate rises, so the verification asymmetry is largest for a weak policy and shrinks as the policy learns (Definition 2.1). Learning closes the asymmetry for the learner, which is the book's thesis in its first and simplest case (§0.3). Experiment 2.1 and Experiment 2.2 measure all three costs in one unit.

1.5 Mapping is harvesting

1:27 To map an intelligence is to harvest solved pairs where checking and building are cheap and finding is dear. A supervised training set is a finite sample from the graph of the map it teaches: pairs (x, y) with y correct for x . A graph can be sampled from either end. From the task end, draw $x \sim \mathcal{D}$ and pay c_{sol} , or a person, to find y . From the answer end, draw y from a prior you control and derive its problem with a cheap forward map: differentiate a function to pose an integral, carve cells out of a filled grid, inject a bug into working code (Definition 2.2). That costs c_{con} , and harvesting means sampling from this end wherever c_{con} is far below c_{sol} . The catch is that an answer-end sample follows the designer's distribution, and the world's differs ((2.2)).

1:28 At inference time a verifier converts compute into intelligence directly: draw n candidates and return one the verifier accepts. Bradley Brown and colleagues ran DeepSeek-Coder-V2-Instruct on SWE-bench Lite, a set of real GitHub issues, and counted an issue solved when any attempt passed its unit tests. With one attempt it solved 15.9%; with 250 it solved 56% (Brown et al., "Large Language Monkeys"). On math problems, where they picked answers without an automatic verifier, majority voting and reward models "plateau beyond several hundred samples." Sampling buys coverage (Proposition 2.3), and soundness decides how much of the coverage is real (Proposition 1.1).

1:29 Every object defined so far has a verifier inside it. (1.1) contains V ; universal search consults a verifier at every step; reinforcement learning needs a reward, and a verifiable reward is a verifier; best-of- n sampling is only as good as its verifier is sound. Backward construction seems to need none, because its pairs are correct by construction, but a constructor is a verifier with privileged access to the answer. Whoever owns a cheap, sound verifier for a task family owns a training signal for it. A verifier is a capital good: built once, as a test suite, a proof kernel, a game engine, a digital twin, a wet lab or a profit-and-loss statement, it

turns compute into capability until the family saturates. Andrej Karpathy's version takes two lines: "Software 1.0 easily automates what you can specify. Software 2.0 easily automates what you can verify" ([Karpathy, "Verifiability"](#)).

1:30 Verifiability is the v_k of the master equation, and (1.1) says why it multiplies the closing rate. Intelligence is defined only relative to a verifier, so intelligence supply aimed at a family with no verifier has nothing to optimize and closes nothing, however large it is (§0.4).

1:31 The product makes the frontier jagged. Karpathy calls verifiability "what's driving the 'jagged' frontier of progress in LLMs": verifiable tasks "progress rapidly," while creative and strategic tasks, and those that combine "real-world knowledge, state, context and common sense," lag ([Karpathy, "Verifiability"](#)). Wei predicted "a jagged edge of intelligence, where AI is much smarter at verifiable tasks" ([Wei, "Asymmetry of verification"](#)). Ranked by their verification asymmetry (Figure 2.3), task families give the order in which the intelligence curve should climb.

1:32 The ranking also shows where the other asymmetries take over. Where the verifier is a human expert, verification inherits human latency and headcount (§5.1; §8.3). Where it is the physical world, it inherits physical inertia (§15.2). Where it is a stored key, verification becomes a security problem (§12.6). I place frontier AI research near the judgment end of the ranking, so the research loop runs through one of the hardest families to check (§11.1; §11.5).

1:33 If intelligence is a curve, evaluations should report curves. ARC Prize already requires them: "Due to variable inference budget, efficiency (e.g., compute cost) is now a required metric when reporting performance," it wrote when it published o3's two scores ([ARC Prize](#)). A single headline number is easier to sell than a curve, so I put its spread to model launches at two in five.

1:34

Forecast 1.1 (Intelligence reported as a curve). By the end of 2028, frontier model launches report their headline evaluations as curves, each score tied to the compute, cost or reasoning budget that produced it, and models are compared by where their curves cross.

Horizon 2028-12-31

Probability 40%

Check Take the latest flagship launch of OpenAI, Anthropic, Google DeepMind and xAI on 31 December 2028. The forecast holds if at least three of the four report most headline benchmark results at three or more stated budgets, or as plots of score against cost or compute. It fails if most headline results are still single scores with no budget attached.

1:35 If the soundness a verifier needs grows with capability, so does the cost of keeping it. OpenAI's August 2026 post on pacing model development put numbers on that cost: "a two-week pause in reinforcement learning (RL) training on our latest models intended for deployment while we further hardened and red-teamed our research environments," and monitoring that costs "roughly 20% of the inference compute being monitored" ([OpenAI, "Pacing model development in an era of cyber-critical capabilities"](#)). I take those as the baseline as of October 2026. The argument for the forecast is strong and labs disclose little, so I put it a little under even.

1:36

Forecast 1.2 (Verifier hardening grows with capability). By the end of 2028, a frontier lab discloses that hardening verifiers (red-teaming environments, isolating answer keys, monitoring runs) costs more than the roughly 20% of monitored inference compute that OpenAI reported in August 2026, as the cost of keeping verifiers sound rises with capability.

Horizon 2028-12-31

Probability 45%

Check Read the frontier labs' system cards, safety reports and engineering posts published through 2028. The forecast holds if a lab discloses a hardening or monitoring overhead above 20% of the compute it hardens or monitors; it fails otherwise.

1:37 Judgment still lacks a cheap sound verifier (2:46). I put the jagged order at about two in three through 2028. The likeliest way it fails is transfer: skill learned on verifiable work spilling into judgment without a verifier of its own, which would add a term the

master equation lacks.

1:38

Forecast 1.3 (The frontier stays jagged). Through 2028, capability gains keep the order of verifiability: formal mathematics and tested code improve fastest, open-ended judgment (long-form writing, strategy, research taste) slowest, and the distance between them widens while judgment lacks a cheap sound verifier.

Horizon 2028-12-31

Probability 65%

Check Take the hidden-test code benchmark and the expert-graded judgment benchmark with the most frontier-model results at both dates, such as SWE-bench Pro and GDPval, and compare the share of remaining headroom each closed from the end of 2026 to the end of 2028, counting headroom up to the share of tasks its maintainers have verified as solvable. The forecast fails if the judgment benchmark closed as large a share or larger without first gaining an automated grader that agrees with its human experts about as often as they agree with each other; it holds otherwise.

References

1. ARC Prize, "OpenAI o3 Breakthrough High Score on ARC-AGI-Pub". <https://arcprize.org/blog/oai-o3-pub-breakthrough>
2. Wei, "Asymmetry of verification and verifier's law". <https://www.jasonwei.net/blog/asymmetry-of-verification-and-verifiers-law>
3. Shane Legg, Marcus Hutter (2007). Universal Intelligence: A Definition of Machine Intelligence. *arXiv*. <https://arxiv.org/abs/0712.3329>
4. François Chollet (2019). On the Measure of Intelligence. *arXiv*. <https://arxiv.org/abs/1911.01547>
5. ARC Prize, "2024 Progress on ARC-AGI-Pub". <https://arcprize.org/blog/2024-progress-arc-agi-pub>
6. Levin 1973. https://www.mathnet.ru/php/archive.phtml?jmid=ppi&option_lang=eng&paperid=914&wshow=paper
7. Scholarpedia, "Universal search". http://www.scholarpedia.org/article/Universal_search
8. Grégoire Delétang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein et al. (2023). Language Modeling Is Compression. *arXiv*. <https://arxiv.org/abs/2309.10668>
9. Strathern, "Improving ratings: audit in the British University system". <https://www.cambridge.org/core/journals/european-review/article/improving-ratings-audit-in-the-british-university-system/FC2EE640C0C44E3DB87C29FB666E9AAB>
10. Epoch AI, "An FAQ on Reinforcement Learning Environments". <https://epoch.ai/gradient-updates/state-of-rl-envs>
11. Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le et al. (2024). Large Language Monkeys: Scaling Inference Compute with Repeated Sampling. *arXiv*. <https://arxiv.org/abs/2407.21787>
12. Karpathy, "Verifiability". <https://karpathy.bearblog.dev/verifiability/>
13. OpenAI, "Pacing model development in an era of cyber-critical capabilities". <https://openai.com/index/pacing-model-development-cyber-capabilities/>