IT TAKES TWO: CO-EVOLVING SOCIAL WORLDS AND AGENT ARCHITECTURES

Anonymous authors
Paper under double-blind review

ABSTRACT

The worlds in which large language model agents operate are becoming increasingly complex and open-ended, placing growing demands on the architectures that organize their reasoning, memory, and action. However, agent architectures are still typically designed or optimized in fixed worlds that quickly saturate, weakening the evolutionary pressure needed to discover stronger structures and strategies. We propose WoArc, a world–architecture co-evolution framework that makes the world itself an adaptive part of agent design, allowing worlds to evolve around the limitations of current architectures and architectures to evolve against the resulting challenges. WoArc closes this loop by letting current architectures guide the search for informative new worlds, accepting challenges through method-agnostic criteria, and feeding them back into executable architecture evolution, while a resolution-calibrated controller decides whether to keep evolving architectures, refresh the worlds, or stop. Experiments on hidden-information social interactions show that world evolution restores meaningful discrimination among increasingly capable agents, while the resulting architectures outperform published alternatives on test interactions. Our code and the evolved benchmark are available at `https://anonymous.4open.science/r/WoArc-29CC`.

1 INTRODUCTION

The worlds in which language model agents operate are becoming increasingly complex and open ended, placing growing demands on the architectures that organize their reasoning, memory, and action. This challenge is particularly evident in social interaction, where agents must act over multiple turns while reasoning about private goals and incomplete information. Interactive benchmarks such as SOTOPIA (Zhou et al., 2024b) have made these abilities measurable, while a growing body of training free methods targets different aspects of social reasoning (Hwang et al., 2026; Kobalczyk et al., 2025; Hu et al., 2024; Suri et al., 2026; Luo et al., 2026). As these worlds become more demanding, the design of agent architectures becomes increasingly important.

Agent architectures, however, are still typically designed or searched on fixed worlds. Whether a scaffold is manually designed or discovered through automated search (Hu et al., 2025; Zhang et al., 2025b;a), it is usually evaluated on a fixed set of scenarios. As agents improve, such benchmarks can become saturated and provide less discrimination among competing methods. We observe this effect on our initial hidden fact benchmark of 16 human authored worlds, where Vanilla already achieves a success rate of 0.7891 and six published method variants span only 0.1641. When most methods perform within a narrow range, the benchmark provides a weak signal for distinguishing better architectures. Architecture search then faces the same limitation, since improvements that the benchmark cannot distinguish cannot be reliably selected.

Figure 1 illustrates why benchmark quality matters for architecture search. On a fixed benchmark, candidate architectures may receive increasingly similar evaluation results as the benchmark becomes saturated, making it difficult for search to identify useful improvements. In the example, the resulting agent asks an uninformative question, remains uncertain about the partner's preference, and makes the wrong decision. WoArc instead evolves worlds together with architectures, continually introducing worlds that reveal differences not captured by the current benchmark. These more discriminative worlds provide a more informative basis for architecture search, allowing the architecture to develop more effective elicitation and decision making strategies.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

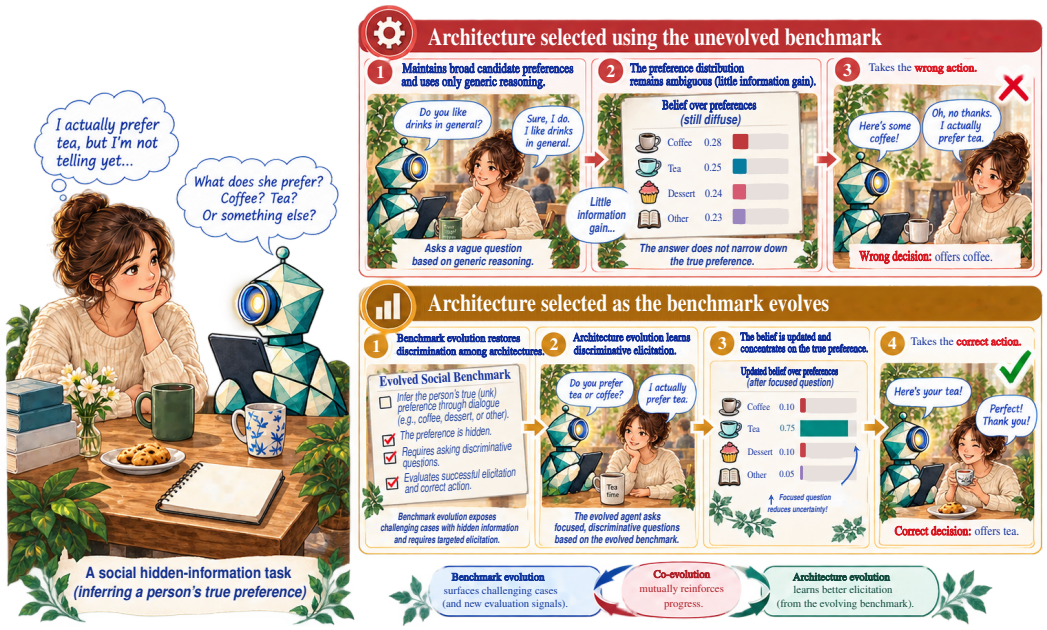

Figure 1: A hidden-fact social interaction. The partner's true preference is never stated; a generic scaffold asks an undiscriminating question, keeps a diffuse belief and acts on the wrong one, while the architecture WoArc evolves asks a discriminating question and commits to the right action.

We propose WoArc, a framework that jointly evolves hidden-fact worlds and agent architectures in a single loop (Figure 2). On the *world side*, an LLM generates candidate worlds through architecture-aware generation, which tells the generator which mechanism the current architectures rely on, and architecture-independent generation, which does not,, while acceptance is determined only by two fixed reference agents, Vanilla and Oracle. On the *architecture side*, agents are represented as executable graphs and evolved through typed structural edits, causal screening, and family-wise Pareto selection. A resolution-calibrated controller determines whether to continue architecture evolution, evolve the worlds, or stop. Starting from Vanilla, three generations of WoArc achieve a gain of $+0.1400$ on 400 test episodes, outperforming the other published baselines, while one round of world evolution increases the spread among evaluated methods from 0.1641 to 0.2941. Our contributions are threefold:

- **World–architecture co-evolution.** We introduce a framework that jointly evolves hidden-fact worlds and executable agent architectures. World evolution combines architecture-aware and architecture-independent generation with method-independent acceptance, allowing the benchmark to remain informative as architectures improve.
- **Architecture evolution with structured search and adaptive control.** We evolve agent architectures through typed graph edits, causal screening, and family-wise Pareto selection. A resolution-calibrated controller determines whether to continue architecture search, evolve the worlds, or stop.
- **Empirical validation of both world and architecture evolution.** One round of world evolution increases the spread among evaluated methods from 0.1641 to 0.2941 and improves ten of thirteen benchmark-quality measures. Starting from Vanilla, three generations of architecture evolution achieve a gain of $+0.1400$ on 400 test episodes.

2 Related Work

2.1 Benchmarks for social agents

SOTOPIA (Zhou et al., 2024b) evaluates agents pursuing private social goals and already needed a hard subset on which every model struggles; simulated interaction also looks far more successful

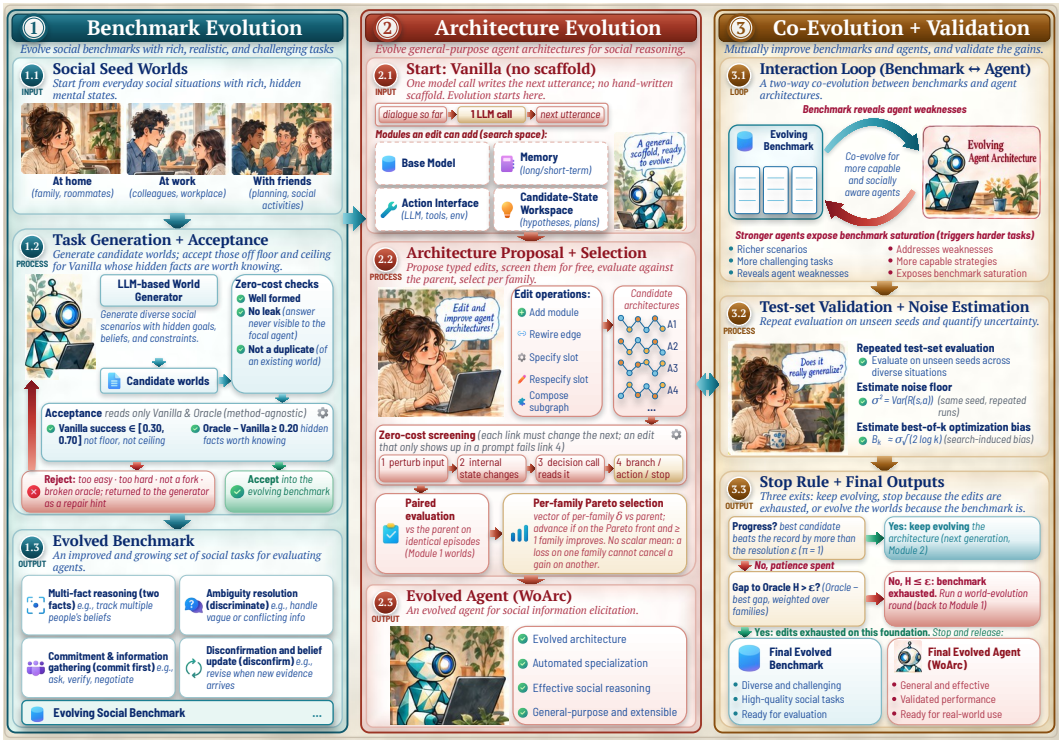

Figure 2: WoArc evolves hidden-fact worlds and agent architectures in one loop. Worlds come from architecture-aware and architecture-independent generation, and are accepted only if Vanilla succeeds at a rate in $[0.30, 0.70]$ and Oracle gains at least 0.20—a test that never consults the architecture under test.

when one model sees everyone's information than under realistic asymmetry (Zhou et al., 2024a). Dynamic benchmarks stay ahead of models through human-and-model-in-the-loop authoring (Kiela et al., 2021), LLM reframing of existing instances (Wang et al., 2025), or optimisation of declared desiderata such as difficulty (Li et al., 2025b). In each case the model under test decides what enters, so the benchmark becomes hard for one system and says nothing about the next. WoArc hardens it against a population instead: the acceptance criterion never consults the architecture under test, only the fixed Vanilla and Oracle references.

2.2 Eliciting hidden information

Training-free methods each specialise in one part of the elicitation problem: modelling the partner between turns (Hwang et al., 2026; Cross et al., 2025; Zhang et al., 2025c; Xia et al., 2025), choosing the question by expected information gain (Kobalczyk et al., 2025; Choudhury et al., 2026; Hu et al., 2024), and deciding when to stop asking (Li et al., 2024; Suri et al., 2026; Luo et al., 2026; Liu et al., 2025). Compiled into one slot space (Appendix B.9), they occupy only a handful of its slots. Each is a hand-designed point in that space, fixed at publication; WoArc searches the space and, from an empty agent, reaches a point that outperforms all of them (§4.3).

2.3 Architecture search and environment generation

Agents have been optimised as computational graphs (Zhuge et al., 2024), programmed by a meta agent (Hu et al., 2025), searched over code workflows (Zhang et al., 2025b), evolved over a modular design space (Shang et al., 2025), sampled from an agentic supernet (Zhang et al., 2025a), and grown as self-modifying coding agents (Zhang et al., 2026). Conversely, POET (Wang et al., 2019), PAIRED (Dennis et al., 2021) and ACCEL (Parker-Holder et al., 2023) evolve environments at the frontier of an agent's ability, foundation models now write the environments themselves (Faldor et al., 2025; Zala et al., 2024; Zhou et al., 2025; Gu et al., 2026), and POBAX (Tao et al., 2025),

closest to our acceptance criterion, requires a large gap between agents with more and less state information. The former cannot select for what its fixed benchmark cannot separate; the latter treats the world as scaffolding for one policy and discards it. WoArc evolves both and keeps the coupling auditable.

3 METHOD

3.1 BENCHMARK EVOLUTION

A world $w = (s, g_P, \mathcal{C})$ pairs a public SOTOPIA scenario (Zhou et al., 2024b) with a private partner goal g_P and K *hidden facts* $\mathcal{C} = \{(c_k, v_k, \mathcal{A}_k)\}$: the correct value v_k of coordinate c_k among plausible decoys $\mathcal{A}_k$, plus a rule for when v_k is disclosed. An episode is a $T = 20$ turn conversation between the *focal agent* under evaluation and a *partner*, an unscaffolded LLM that holds g_P and $\mathcal{C}$ and never shows them. Worlds come in four families: **two facts** (conditions disclosed only when raised separately), **discriminate** (two readings that generic questions cannot separate), **commit first** (the condition is named only after a costly commitment), and **disconfirm** (an early literally-true statement suggests the wrong need).

A generator turns an unused scenario s and family f into a candidate world in one of two modes that differ by a single prompt suffix: *architecture-independent generation* $w_B \sim G_\phi(\cdot \mid p_B(s, f))$ and *architecture-aware generation* $w_A \sim G_\phi(\cdot \mid p_B(s, f) \oplus m)$, where the note m names the mechanism current architectures rely on and forbids making worlds easier for it. Scoring every method on the two halves separately turns any tilt towards that mechanism into a measured gap: *circularity becomes a number we report rather than an assumption we defend.* Malformed, duplicate or answer-leaking candidates are discarded at zero cost.

Every measurement in this paper is a success rate: a judge decides per fact, from a verbatim quote, whether the focal agent's *own* lines state the correct value (majority of three), and an episode counts only if every fact is recovered,

$$y(\pi, w, \sigma) \;=\; \prod_{k=1}^{K} \mathbb{1}[\,\text{focal agent states } v_k\,], \tag{1}$$

with $S(a; w)$ the mean of y over seeds σ. Two reference agents calibrate every world: Vanilla, an LLM agent with no scaffold, and Oracle, Vanilla with the hidden facts in its context, an upper bound on what knowing them is worth. A candidate is evaluated with $n = 8$ seeds of each and accepted iff

$$S(Vanilla; w) \in [0.30,\ 0.70] \quad \text{and} \quad S(Oracle; w) - S(Vanilla; w) \;\geq\; 0.20, \tag{2}$$

which keeps the world off floor and ceiling and requires the hidden facts to be worth knowing. Equation 2 reads only w, Vanilla and Oracle: the architecture under test can influence *where the generator looks*, never *what is accepted*. An evolved benchmark replaces the old one when it lowers every method's score while widening the spread between methods,

$$S(a; \mathcal{W}_{\text{old}}) - S(a; \mathcal{W}^\star) > 0 \;\; \forall a \in \mathcal{M}, \qquad \text{spread}(\mathcal{W}^\star) > \text{spread}(\mathcal{W}_{\text{old}}), \tag{3}$$

and, because a merely broken benchmark would also lower every score, we also report its quality on the axes of Figure 3 (Appendix B).

3.2 ARCHITECTURE EVOLUTION

An architecture is an executable graph $a = (V, E, \Theta, \Omega)$ of typed nodes (observation, hypothesis update, planning, stop-or-continue, memory, deterministic update, action) joined by guarded edges, with Ω the output fields of each model call; model calls follow from the graph, so a graph without a stop-or-continue field cannot stop.

Search proposes *typed edits* from a vocabulary of eleven edit types that create *interfaces but no content*: an open output schema, a planning call, a memory channel, a deterministic update, a control branch and its condition, a consumer edge, parallel observers, a verifier, a hypothesis set, and slot re-specification. No edit type says when to ask, stop, commit or concede: *whatever social strategy appears in an evolved agent was found by search, not written by us.* A candidate chains two or three

edit types and is simulated only after a zero-cost screen certifies a causal chain from a perturbed input to a changed branch, action or stopping condition. Survivors are evaluated against their parent on identical episodes and scored by the vector of paired per-family differences

$$\boldsymbol{\delta}(a \mid p) = \big(\delta_f\big)_{f\in\mathcal{F}}, \qquad \delta_f = \frac{1}{|\mathcal{K}_f|}\sum_{(w,\sigma)\in\mathcal{K}_f}\big[y(a,w,\sigma) - y(p,w,\sigma)\big]. \tag{4}$$

A candidate advances if it lies on the generation's Pareto front and improves at least one family; a scalar average would let a loss on one family cancel a gain on another.

3.3 A RESOLUTION-CALIBRATED STOPPING RULE

Prior agent and environment design systems run for a fixed number of iterations (Hu et al., 2025; Zhang et al., 2025b; 2026; Faldor et al., 2025); we calibrate stopping on the evaluation itself. Let $\hat{S}_n(a) = S(a) + \xi_a$ with $\xi_a \sim \mathcal{N}(0, \hat{q}/2n)$, where $\hat{q}$ is the disagreement rate between two independent evaluations of one architecture, measured from repeated Vanilla runs. A generation keeps the best of m candidates, so the winner exceeds its parent by chance alone; its resolution is the $(1-\alpha)$-quantile of this winner's curse,

$$\varepsilon_m(n) \;=\; \inf\Big\{x:\ \Pr\big[\max_{i\le m}\xi_i - \xi_0 \le x\big] \ge 1-\alpha\Big\}. \tag{5}$$

With G_t the candidates of generation t and R_{t-1} the record, generation t makes progress if $\pi_t = \mathbb{1}[\max_{a\in G_t}\hat{S}_{n_t}(a) - R_{t-1} > \varepsilon_{m_t}(n_t)]$, and k_t counts consecutive stalls. Patience follows evidence rather than a fixed count: with g the smallest gain worth one more generation and $\beta_t(g) = 1 - \Phi\big((\varepsilon_{m_t}(n_t) - g)/\sqrt{\hat{q}/n_t}\big)$ the power of generation t against it, evolution stops at

$$\tau \;=\; \min\Big\{t:\ \prod_{s=t-k_t+1}^{t}\big(1-\beta_s(g)\big) \le \delta\Big\}. \tag{6}$$

Proposition 1. *Under this noise model, (i) a generation in which no candidate improves on the record registers progress with probability at most α; (ii) if every generation after t_0 contains a candidate with true gain at least g, the rule stops after t_0 with probability at most δ.* Proof and constants are in Appendix D.

Progress can also stall because the worlds leave nothing to gain: with w_f the share of family f and $a_t^\star$ the current best architecture, the remaining gap to Oracle is

$$H_t \;=\; \sum_f w_f\,\big(S_f(Oracle) - S_f(a_t^\star)\big). \tag{7}$$

If $\pi_t = 1$, WoArc keeps evolving architectures; if Eq. 6 fires while $H_t > \varepsilon_{m_t}(n_t)$, the edit vocabulary is exhausted on this foundation; if $H_t \le \varepsilon_{m_t}(n_t)$, the benchmark is exhausted and a world-evolution round is run, its result accepted only if it restores a gap to Oracle without narrowing the spread of Eq. 3.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Benchmarks. The *unevolved* benchmark is 16 human-authored worlds, four per family (128 episodes). One round of world evolution proposes 104 worlds from unused SOTOPIA scenarios, of which the acceptance criterion (Eq. 2) keeps 17 across all four families (10 from architecture-aware and 7 from architecture-independent generation), giving the *evolved* benchmark (136 episodes). The test set is four unseen worlds, one per family, from the evolved benchmark, crossed with seeds 17–116 (400 episodes per architecture).

Table 1: success rate before and after one round of world evolution: the unevolved benchmark (16 worlds) versus the evolved benchmark (17 worlds, split by generation mode). Each published method runs on two host architectures, Seed Architecture and Probe-then-Act.

	Before	After (evolved benchmark)			
Method	Unevolved	Architecture-aware	Architecture-independent	Merged	Drop
Vanilla (no scaffold)	0.7891	0.5375	0.4464	0.5000	+0.2891
ToMA + Seed Architecture	0.8203	0.8250	0.7500	0.7941	+0.0262
ToMA + Probe-then-Act	0.7109	0.7000	0.4107	0.5809	+0.1300
CONVOLVE + Seed Architecture	0.8516	0.7875	0.6964	0.7500	+0.1016
CONVOLVE + Probe-then-Act	0.7891	0.5750	0.6250	0.5956	+0.1935
ATD + Seed Architecture	0.8359	0.7875	0.6786	0.7427	+0.0932
ATD + Probe-then-Act	0.6875	0.6625	0.6071	0.6397	+0.0478
Spread (max−min), all rows	0.1641			0.2941	
Spread, published methods only	0.1641			0.2132	

Baselines. We compare against ten training-free published methods. Five elicit or model the partner's hidden state: ToMA (Hwang et al., 2026), MediQ (Li et al., 2024), ATD (Kobalczyk et al., 2025), CONVOLVE (Luo et al., 2026) and GATE (Li et al., 2025a). Four are general reasoning scaffolds: zero-shot CoT (Kojima et al., 2022), ReAct (Yao et al., 2023), ReflAct (Kim et al., 2025) and Plan-and-Solve (Wang et al., 2023). One is the negotiation agent LLM-Deliberation (Abdelnabi et al., 2024). Each is implemented on Vanilla with its published prompts, under the same budget of at most two model calls per turn as our architectures, and audited against its source (Appendix A).

Details. Agents, partners and judges all use `deepseek-v3.2`; worlds are proposed by `deepseek-v3`. Every score is a three-vote majority judgement, and an episode succeeds only if every hidden fact is recovered. Architecture search sees only a development set; the test worlds and seeds are used once (§4.3). WoArc is the champion of three generations of search from Vanilla (8, 12 and 11 candidates, each on 80 paired episodes); its lineage adds a composed-mechanism, an aggregation and a multi-call-composition edit.

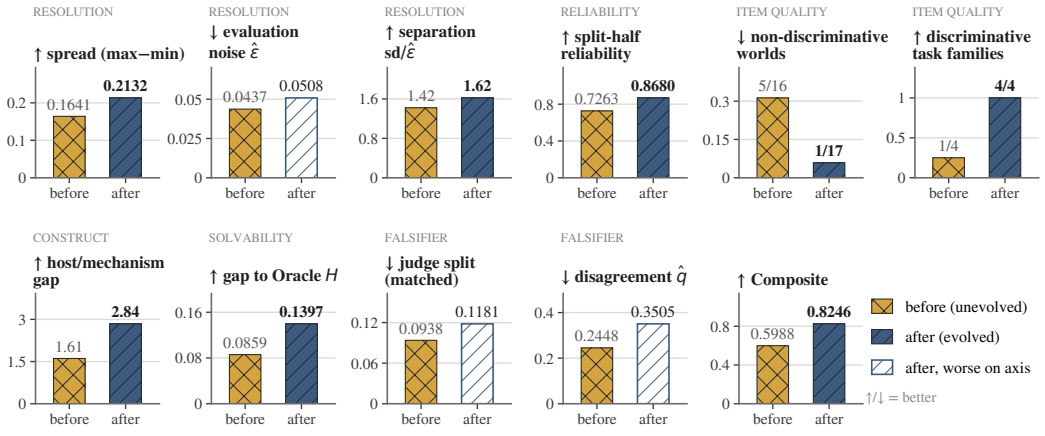

Figure 3: Benchmark quality before and after world evolution, over the six published-method variants. Each panel has its own scale; arrows give the direction of improvement, hollow bars mark axes on which the evolved benchmark is worse, and the composite is an equal-weight mean of six components.

Table 2: Test-set success rate on 400 episodes. Δ is the paired difference against Vanilla (0.5425), overall and per family.

Method	Δ vs Vanilla	Two facts	Discri-minate	Commit first	Discon-firm	Success
Vanilla (no scaffold)	0	–	–	–	–	0.5425
Published methods, as published, on Vanilla						
GATE, yes/no questions (Li et al., 2025a)	+0.1400	−0.15	+0.18	+0.10	+0.43	0.6825
ATD (Kobalczyk et al., 2025)	+0.1250	−0.16	+0.21	+0.07	+0.38	0.6675
ToMA (Hwang et al., 2026)	+0.1200	0.00	+0.17	+0.17	+0.14	0.6625
MediQ (Li et al., 2024)	+0.0950	−0.07	+0.12	+0.35	−0.02	0.6375
GATE, open-ended questions (Li et al., 2025a)	+0.0825	−0.17	+0.19	+0.12	+0.19	0.6250
Plan-and-Solve (Wang et al., 2023)	+0.0325	+0.03	−0.04	−0.08	+0.22	0.5750
CONVOLVE (Luo et al., 2026)	+0.0050	−0.05	−0.10	+0.14	+0.03	0.5475
Zero-shot CoT (Kojima et al., 2022)	−0.0175	−0.16	−0.01	−0.08	+0.18	0.5250
ReAct (Yao et al., 2023)	−0.0300	−0.10	−0.04	−0.27	+0.29	0.5125
ReflAct (Kim et al., 2025)	−0.0800	−0.07	−0.08	−0.31	+0.14	0.4625
LLM-Deliberation, greedy (Abdelnabi et al., 2024)	−0.1425	−0.26	−0.02	−0.39	+0.10	0.4000
Architectures evolved by WoArc						
WoArc (ours)	**+0.1400**	−0.07	+0.22	+0.42	−0.01	**0.6825**
WoArc (from Seed Architecture)	+0.2400	−0.25	+0.30	+0.48	+0.43	0.7825

4.2 Does World Evolution Improve Benchmark Discrimination?

The evolved benchmark better separates agent architectures. All seven complete rows of Table 1 score lower on the evolved benchmark, with drops from $+0.0262$ to $+0.2891$, and the spread across agents grows from 0.1641 to 0.2941. This gain does not come mainly from architecture-aware generation. For six of the seven agents evaluated, the worlds generated without any architecture information are in fact the harder half; Vanilla, for example, succeeds at 0.4464 on that half against 0.5375 on the architecture-aware half. The two generation modes are also accepted at similar rates ($10/51$ versus $7/52$).

The improvement is not caused by making worlds unsolvable. Since anything that breaks worlds also lowers every score, we measure the quality of the evolved benchmark (Figure 3): it is better on ten of thirteen directional quantities, the composite rises from 0.5988 to 0.8246, separation in units of its own noise from 1.42 to 1.62, and split-half reliability from 0.7263 to 0.8680. Three of the four families sat above saturation on the unevolved benchmark (discriminate 0.8958, commit first 0.9115, disconfirm 0.9219), collapsing a four-way construct to one; all four now lie between 0.5764 and 0.8125. And the worlds are harder to *elicit*, not harder to solve: Oracle is unchanged ($0.9375 \rightarrow 0.9338$) while Vanilla falls by 0.2891, so the remaining gap to Oracle (H, Eq. 7) grows from 0.0859 to 0.1397.

4.3 Do Evolved Architectures Outperform Published Methods?

The evolved architecture outperforms every published method. WoArc is evolved from Vanilla, the same agent every published row is built on, and reaches $+0.1400$, the best result in the table, ahead of ATD ($+0.1250$), ToMA ($+0.1200$), MediQ ($+0.0950$) and CONVOLVE ($+0.0050$). A search over five worlds and 80 episodes per candidate thus finds a better agent than any published design without a hand-written scaffold to start from. Run instead from the hand-written committee, the identical search reaches $+0.2400$; we report that row as an upper reference on what a starting point is worth, not as our main result. The general reasoning scaffolds (CoT, ReAct, ReflAct, Plan-and-Solve) never beat Vanilla by more than the resolution and all lose on commit first; the methods that gain either keep explicit hypotheses about the partner and act on them (ATD, ToMA) or choose what to ask next (GATE), with their gains on discriminate and disconfirm. WoArc gains most on commit first and discriminate ($+0.42$, $+0.22$) and gives up little on two facts (-0.07).

4.4 Analysis of Evolved Agent Behavior

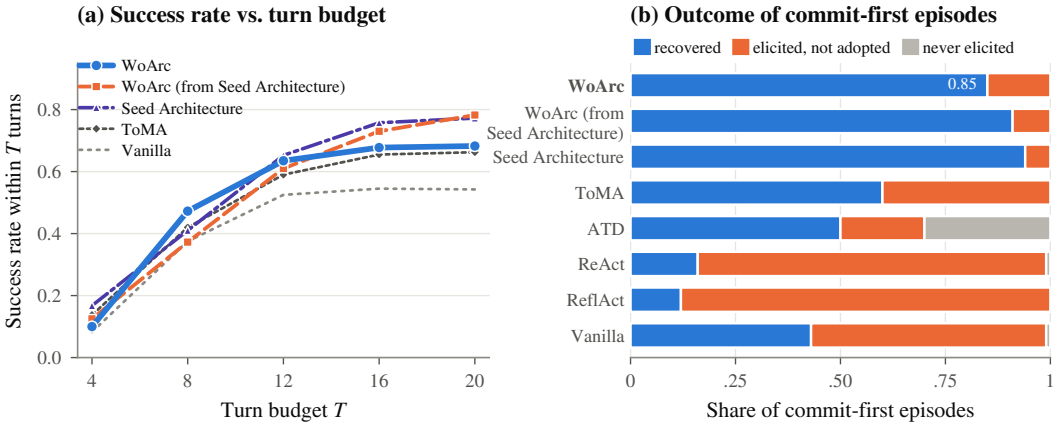

Figure 4: **(a)** Test-set success rate when each transcript is truncated to its first T turns and re-judged. **(b)** Outcome of the hidden condition in the commit-first family: recovered, elicited from the partner but not adopted, or never elicited.

WoArc acquires useful information earlier in the interaction. We truncate each test transcript after the first T turns and re-evaluate success (Figure 4a). At turn 8, WoArc reaches a success rate of 0.4725, compared with 0.4200 for ToMA and 0.3750 for Vanilla. WoArc reaches its final success rate of 0.6825 with an average of 15.1 turns, compared with 15.7 for Vanilla. In contrast, WoArc (from Seed Architecture) reaches only 0.3725 at turn 8 but surpasses the other methods by turn 16 and uses 19.9 turns on average. These results show that higher final success does not necessarily imply faster information acquisition.

Table 3: Transfer to MediQ: 150 cases per agent, no further evolution or tuning, scored by an objective four-way choice.

Agent	Questions/case	Accuracy
Oracle (told the facts)	0.00	0.8133
WoArc (ours)	5.97	**0.6600**
ATD	9.73	0.6467
GATE, open-ended	7.33	0.6467
GATE, yes/no	8.50	0.6400

Methods fail differently on commit-first worlds. Commit-first worlds require an initial commitment before the hidden condition is revealed, followed by correct use of that condition (Figure 4b). ReAct and ReflAct usually elicit the condition but fail to incorporate it, with revealed-but-unused rates of 0.83 and 0.88, versus 0.56 for Vanilla. ATD shows the opposite pattern, failing to trigger revelation in 0.30 of episodes by continuing to ask instead of committing. WoArc recovers the condition in 0.85 of episodes.

Transfer to an external benchmark. To test transfer beyond our generated worlds, we evaluate the evolved architectures on MediQ (Li et al., 2024) without further evolution or tuning (Table 3). MediQ differs in domain, dataset, and evaluation, using an objective four-way choice instead of an LLM judge. Oracle reaches 0.8133 accuracy. WoArc performs best among the evaluated agents at 0.6600, compared with 0.6467 for ATD, 0.6467 for GATE with open-ended questions, and 0.6400 for GATE with yes/no questions. It also uses the fewest questions, averaging 5.97 per case versus 7.33–9.73 for the other methods.

4.5 Ablation Study

World evolution ablations. The two generation modes are complementary (Figure 5). Architecture-aware generation produces acceptable two-facts worlds but leaves the disconfirm family saturated at 0.8958, whereas architecture-independent generation reduces disconfirm to 0.6042 but produces no acceptable two-facts worlds. Using both gives higher resolution, reliability, and composite benchmark quality than either mode alone. The acceptance criterion is also necessary: among 103 valid proposals, 45 are too easy and 36 too hard for Vanilla.

Architecture evolution ablations. The main performance gain comes from the evolved architecture edits. Test success increases from 0.5425 to 0.6650, 0.6675, and finally 0.6825 as the edits

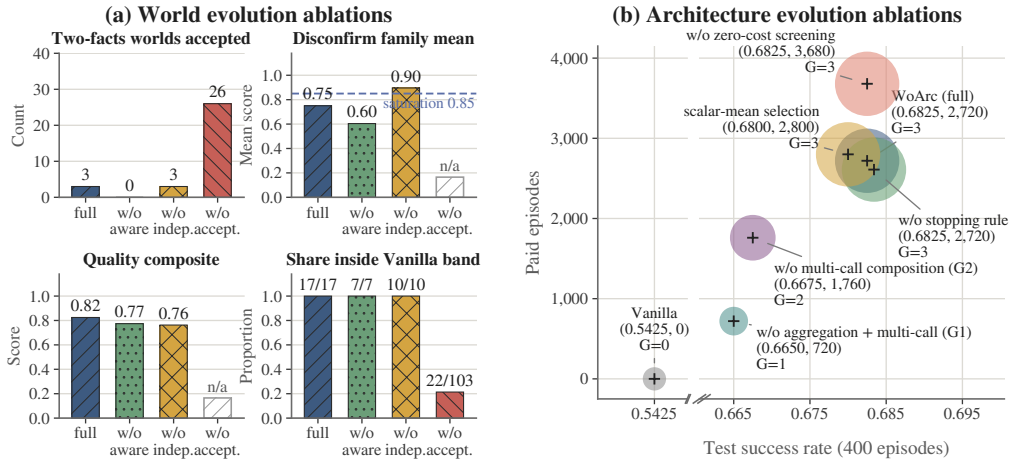

Figure 5: Ablation of WoArc. *(a)* One round of world evolution with one component removed. *(b)* Architecture evolution from Vanilla: test success rate against paid evaluation episodes, with bubble size giving the generation at which the search stops (the horizontal axis is broken to include Vanilla).

Table 4: Comparison of the evolved benchmark W_1 and W_2 obtained after one further round of world evolution.

Metric	W_1	W_2
Vanilla success ↓	0.5000	**0.4643**
Mean of published variants ↓	0.6838	**0.5762**
Oracle success ↑	0.9338	**0.9857**
Method spread ↑	0.2132	**0.2929**
Signal-to-noise ratio ↑	1.615	**1.949**

are added. Replacing family-wise Pareto selection with scalar-mean selection gives a similar score (0.6800), indicating that the gain mainly comes from the discovered edits. Zero-cost screening reduces evaluation from 3,680 to 2,720 episodes. The stopping rule also avoids premature termination: the first two generations remain below the evaluation resolution, while the third exceeds it ($+0.2125$ versus 0.1823).

4.6 Additional Analysis: Continuing the Co-Evolution Loop

Starting from the evolved benchmark used for architecture search, denoted as W_1, we perform one further round of world evolution to obtain W_2. Compared with W_1, W_2 is harder for Vanilla and published methods while remaining highly solvable by Oracle; the spread among six published method variants increases from 0.2132 to 0.2929, and the signal-to-noise ratio from 1.615 to 1.949 (Table 4). Further architecture evolution, however, finds no valid successor under the current edit operators, suggesting that the bottleneck has shifted from benchmark discrimination to the local architecture search neighborhood. Broader or adaptive structural edits may enable further progress.

5 Conclusion

WoArc addresses the weakening selection signal of fixed benchmarks by co-evolving hidden-information social worlds and executable agent architectures. World evolution combines architecture-aware and architecture-independent generation with method-independent admission, while architecture evolution uses typed edits, causal screening, and resolution-calibrated control. Experiments show that world evolution restores method discrimination, while three architecture generations from Vanilla yield a $+0.1400$ gain on 400 test episodes. Together, these results show that jointly adapting worlds and architectures provides a stronger basis for agent design.

AI USE STATEMENT

In this work, we did not use generative AI tools for any of the tasks that require disclosure: generating synthetic datasets, developing theoretical models or conceptual frameworks, formulating mathematical claims or proofs, proposing or refining hypotheses, designing or evaluating the experimental methodology, implementing methods, translation, cleaning or reformatting data, qualitative or thematic analysis, or interpreting results. Generative AI tools were used for two purposes only. First, to aid and polish the writing: the text was written by the authors, and AI tools were used to improve its readability, grammar and phrasing; all technical content, claims and numbers were written and verified by the authors. Second, to produce the decorative artwork (characters and scene elements) in the illustrations of Figures 1 and 2; the layout, the text and the results these illustrations depict were designed and checked by the authors, and no data figure or table in the paper was generated by AI. Neither use affects the methods, experiments or conclusions of the paper. We have reviewed all AI-assisted work and take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Sahar Abdelnabi, Amr Gomaa, Sarath Sivaprasad, Lea Schönherr, and Mario Fritz. Cooperation, competition, and maliciousness: Llm-stakeholders interactive negotiation, 2024. URL https://arxiv.org/abs/2309.17234.

Deepro Choudhury, Sinead Williamson, Adam Golinski, Ning Miao, Freddie Bickford Smith, Michael Kirchhof, Yizhe Zhang, and Tom Rainforth. BED-LLM: Intelligent information gathering with LLMs and bayesian experimental design. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=qyylZMLYT8.

Logan Cross, Violet Xiang, Agam Bhatia, Daniel LK Yamins, and Nick Haber. Hypothetical minds: Scaffolding theory of mind for multi-agent tasks with large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=otW0TJOUYF.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design, 2021. URL https://arxiv.org/abs/2012.02096.

Maxence Faldor, Jenny Zhang, Antoine Cully, and Jeff Clune. OMNI-EPIC: Open-endedness via models of human notions of interestingness with environments programmed in code. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Y1XkzMJpPd.

Zhengyao Gu, Jonathan Light, Raul Astudillo, Ziyu Ye, Langzhou He, Wei Cheng, Santiago Paternain, Philip S. Yu, and Yisong Yue. Actor-curator: Co-adaptive curricula via policy-improvement bandits for post-training. In *ICLR 2026 Workshop on Logical Reasoning of Large Language Models*, 2026. URL https://openreview.net/forum?id=2eWesVwzdQ.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems, 2025. URL https://arxiv.org/abs/2408.08435.

Zhiyuan Hu, Chumin Liu, Xidong Feng, Yilun Zhao, See-Kiong Ng, Anh Tuan Luu, Junxian He, Pang Wei Koh, and Bryan Hooi. Uncertainty of thoughts: Uncertainty-aware planning enhances information seeking in LLMs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=CVpuVe1N22.

EunJeong Hwang, Yuwei Yin, Giuseppe Carenini, Peter West, and Vered Shwartz. Infusing theory of mind into socially intelligent llm agents, 2026. URL https://arxiv.org/abs/2509.22887.

Douwe Kiela, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, Zhiyi Ma, Tristan Thrush, Sebastian

Riedel, Zeerak Waseem, Pontus Stenetorp, Robin Jia, Mohit Bansal, Christopher Potts, and Adina Williams. Dynabench: Rethinking benchmarking in nlp, 2021. URL https://arxiv.org/abs/2104.14337.

Jeonghye Kim, Sojeong Rhee, Minbeom Kim, Dohyung Kim, Sangmook Lee, Youngchul Sung, and Kyomin Jung. Reflact: World-grounded decision making in llm agents via goal-state reflection, 2025. URL https://arxiv.org/abs/2505.15182.

Katarzyna Kobalczyk, Nicolas Astorga, Tennison Liu, and Mihaela van der Schaar. Active task disambiguation with llms, 2025. URL https://arxiv.org/abs/2502.04485.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Belinda Z. Li, Alex Tamkin, Noah Goodman, and Jacob Andreas. Eliciting human preferences with language models. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=LvDwwAgMEW.

Shuyue Stella Li, Vidhisha Balachandran, Shangbin Feng, Jonathan S. Ilgen, Emma Pierson, Pang Wei Koh, and Yulia Tsvetkov. Mediq: Question-asking LLMs and a benchmark for reliable interactive clinical reasoning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=W4pIBQ7bAI.

Xiang Lisa Li, Farzaan Kaiyom, Evan Zheran Liu, Yifan Mai, Percy Liang, and Tatsunori Hashimoto. Autobencher: Towards declarative benchmark construction, 2025b. URL https://arxiv.org/abs/2407.08351.

Grace Liu, Yuxiao Qu, Jeff Schneider, Aarti Singh, and Aviral Kumar. Cart: Teaching llm agents to know when they know enough, 2025. URL https://arxiv.org/abs/2510.08517.

Han Luo, Bingbing Wen, and Lucy Lu Wang. Agentic abstention: Do agents know when to stop instead of act?, 2026. URL https://arxiv.org/abs/2606.28733.

Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design, 2023. URL https://arxiv.org/abs/2203.01302.

Yu Shang, Yu Li, Keyu Zhao, Likai Ma, Jiahe Liu, Fengli Xu, and Yong Li. Agentsquare: Automatic LLM agent search in modular design space. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=mPdmDYIQ7f.

Manan Suri, Puneet Mathur, Nedim Lipka, Franck Dernoncourt, Ryan A. Rossi, and Dinesh Manocha. Structured uncertainty guided clarification for llm agents, 2026. URL https://arxiv.org/abs/2511.08798.

Ruo Yu Tao, Kaicheng Guo, Cameron Allen, and George Konidaris. Benchmarking partial observability in reinforcement learning with a suite of memory-improvable domains, 2025. URL https://arxiv.org/abs/2508.00046.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models, 2023. URL https://arxiv.org/abs/2305.04091.

Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O. Stanley. Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions, 2019. URL https://arxiv.org/abs/1901.01753.

Siyuan Wang, Zhuohan Long, Zhihao Fan, Xuanjing Huang, and Zhongyu Wei. Benchmark self-evolving: A multi-agent framework for dynamic LLM evaluation. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert (eds.), *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 3310–3328, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.223/.

Feifan Xia, Yuyang Fang, Defang Li, Yantong Xie, Weikang Li, Yang Li, Deguo Xia, and Jizhou Huang. Probabilistic modeling of intentions in socially intelligent llm agents, 2025. URL https://arxiv.org/abs/2510.18476.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. URL https://arxiv.org/abs/2210.03629.

Abhay Zala, Jaemin Cho, Han Lin, Jaehong Yoon, and Mohit Bansal. Envgen: Generating and adapting environments via llms for training embodied agents, 2024. URL https://arxiv.org/abs/2403.12014.

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. Multi-agent architecture search via agentic supernet, 2025a. URL https://arxiv.org/abs/2502.04180.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Tjarko Lange, and Jeff Clune. Darwin gödel machine: Open-ended evolution of self-improving agents. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=pUpzQZTvGY.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. Aflow: Automating agentic workflow generation, 2025b. URL https://arxiv.org/abs/2410.10762.

Xuanming Zhang, Yuxuan Chen, Samuel Yeh, and Sharon Li. Metamind: Modeling human social thoughts with metacognitive multi-agent systems. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025c. URL https://openreview.net/forum?id=rGMaZkn1ve.

Xuhui Zhou, Zhe Su, Tiwalayo Eisape, Hyunwoo Kim, and Maarten Sap. Is this the real life? is this just fantasy? the misleading success of simulating social interactions with LLMs. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 21692–21714, Miami, Florida, USA, November 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.1208. URL https://aclanthology.org/2024.emnlp-main.1208/.

Xuhui Zhou, Hao Zhu, Leena Mathur, Ruohong Zhang, Haofei Yu, Zhengyang Qi, Louis-Philippe Morency, Yonatan Bisk, Daniel Fried, Graham Neubig, and Maarten Sap. SOTOPIA: Interactive evaluation for social intelligence in language agents. In *The Twelfth International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=mM7VurbA4r.

Yifei Zhou, Sergey Levine, Jason E Weston, Xian Li, and Sainbayar Sukhbaatar. Self-challenging language model agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=9yusqX9DpR.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. Language agents as optimizable graphs, 2024. URL https://arxiv.org/abs/2402.16823.

APPENDIX OVERVIEW

Appendix A documents how each published method was re-implemented and audited. Appendix B reports the additional experiments and analyses referenced from the main text. Appendix C follows one world and one edit end to end. Appendix D gives the proof, constants and replay of the stopping rule.

A BASELINE IMPLEMENTATIONS

This appendix documents the ten published methods of Table 2, which occupy eleven rows because GATE publishes two question forms and we run both. All are our re-implementations in the executable agent-graph representation of §3; none uses the authors' code. For each method we state what is kept from the source, what the environment forces us to change, and what is lost.

A.1 IMPLEMENTATION DETAILS

As-published implementations (Table 2). Every method is implemented on the Vanilla agent (no scaffold) in the most literal form our environment allows. Prompts keep the published wording, with domain words substituted; decoding parameters are copied from the paper or official code where stated and otherwise set to temperature 0.7. The action interface is identical for every agent in the paper, ours included. Before the final evaluation we audited every implementation line by line against its source and removed all wording that the original does not contain and that could help on our task, including phrases borrowed from our own architectures and parameters tuned on pilot experiments (Appendix A). These are our adaptations of the published methods, not the authors' code.

+Seed Architecture and +Probe-then-Act variants (Table 1). For the benchmark comparison, the three methods available at the time (ToMA, CONVOLVE, ATD) were instead added on top of one of two plan-then-act host architectures: the hypothesis committee (the +Seed Architecture variant) or a probe-then-act planner that names the unknown, why it matters and how to elicit it (the +Probe-then-Act variant). These variants inherit the host's scaffold, so we use them to measure the benchmark and as an ablation, and never as baseline rows in the test-set comparison.

Every agent in both tables, including our own architectures, uses at most two model budget. We do not evaluate Uncertainty of Thoughts (Hu et al., 2024): fitting its published procedure into our budget removes two of its three components.

A.2 RULES

Parent. Every method is built on Vanilla, the agent without a scaffold. A method whose published procedure writes an intermediate product before acting writes exactly that product and nothing else.

Wording. Prompts copy the published prompt or method description, with domain words substituted (the task, the partner, the kind of action). Where the source gives no prompt, we use the shortest instruction that states its method.

Decoding. Parameters of the first call are copied from the paper or the official code where stated; where neither states them, temperature is 0.7 and the length cap is 8000 tokens. The action call is fixed by the environment for every agent in the paper, ours included: one JSON object with an action type and an argument, at most 600 tokens, temperature 0.7.

Passing the intermediate product to the action call. The text written by the first call is inserted into the action prompt immediately before the JSON instruction, preceded by a neutral sentence: for CoT, ReAct and ReflAct, *"Before this turn, this was written down: {plan} Now choose {agent}'s action for this turn."*; for ATD, *"Your planning module has produced this for this turn: {plan} Now choose {agent}'s action for this turn, conditioned on the context and this plan."*; for ToMA, the same sentence with "mental states" in place of "plan".

Audit. Earlier implementations of five of the seven methods contained wording that the source does not contain and that could help on our task, most of it borrowed from our own architectures, and some length caps had been set from pilot runs. Before the evaluations reported in Table 2, three independent audits compared every prompt line with its source; every such addition was removed, every pilot-derived parameter was reset by the decoding rule above, and the affected methods were re-evaluated from scratch. Results of the earlier implementations are discarded and are not reported. The removed additions are listed per method below.

A.3 SUMMARY

Table 5: Implementation summary. *First call*: what the extra call writes (temperature / token cap, with source). *State*: what is carried across turns beyond the dialogue history.

Method	First call writes	Decoding (source)	State
ToMA	one paragraph of the speaker's and partner's mental states (App. E, Fig. 18)	0.7 (App. C.1) / 600*	none
MediQ (Binary+RG)	– (reason and confidence bit inside the action JSON)	action call only	none
CONVOLVE	– (distilled rules in context)	action call only	none
ATD	20 solution samples, 5 yes/no questions, simulated answers, EIG, argmax	0.7 (code) / 8000	none
Zero-shot CoT	reasoning after "Let's think step by step."	0.0 / 128 (App. A.4)	none
ReAct	a thought (zero-shot instruction)	0.7 / 8000	none
ReflAct	a reflection on state vs. goal	0.7 / 8000	none

*ToMA's first call was not updated to the default length cap of the decoding rule and kept a 600-token cap; its output is a single paragraph of five to six sentences.

A.4 PER-METHOD DETAILS

ToMA (Hwang et al., 2026). *Kept.* The prompting-only variant with mental-state generation ("Base+MS" in the original paper), which the paper defines as a two-step prompt with one additional inference-time call: the first call uses the Figure 18 prompt (*"Prepare the ground for your very next utterance"*, the six mental-state dimensions, *"Write one short paragraph (5-6 sentences) in natural prose"*, *"Cover at least three dimensions across both sides"*), and the paragraph is passed to the action call. *Lost.* The training stage (look-ahead data generation and fine-tuning) is not used; the Figure 19 utterance prompt is replaced by the shared action call; one mental state is written per turn. The audit found no additions.

MediQ (Li et al., 2024). *Kept.* The Expert's Binary abstention with rationale generation: the action JSON first states a reason and whether the agent is *"confident to pick the correct move ... factually using the conversation log"*; if not, it must *"Ask ONE SPECIFIC ATOMIC QUESTION"*; if so, it acts under *"Assume that you already have enough information"*. *One call*: the confidence bit fits in the action JSON. *Lost.* The original performs abstention and question generation as separate calls (at self-consistency 1) and branches in code, whereas ours branches inside the model; we use Binary rather than the paper's best Scale variant; the action call's temperature is the environment's 0.7 rather than the paper's 0.5. *Removed in the audit*: "about {partner}" in the feature-listing sentence, "knows enough about {partner}" in the confidence test, and an instruction to make "the concrete move ... not another question", copied from one of our architectures.

CONVOLVE (Luo et al., 2026). *Kept.* Stopping rules distilled from trajectories and placed in the context of the action call, with no extra call and no separate stop output. *Distillation*: eight Vanilla trajectories from the training split of our worlds, two per family, the longest episode per scenario, excluding the development set; one distillation call (temperature 0.3) writes the rules. None of these scenarios appears in the test set. *Lost.* Our distillation is a single call with our own prompt (at most 200 words, transcripts cut to 14 turns), much smaller than the original pipeline, and there is no multi-step answer/abstain/gather loop. The audit found no additions.

ATD (Kobalczyk et al., 2025). *Kept.* The published prompts with domain words substituted: *"generate a carefully selected, diverse, and representative set of 20 possible interpretations"*, *"generate the 5 most informative yes/no questions"*, simulate each answer on each sample, compute $\text{EIG} = -\sum p \log p$ over the answer split, and ask the argmax. The action call receives the selected question. *Two calls*: the intermediate products take a median of 997 tokens in pilot runs, above the 600-token action cap; writing none of them leaves a single candidate question, which is the paper's own implicit baseline. *Lost.* The $1+1+N\times M$ calls of the original are merged into one, so samples are not independent; EIG is written by the model; the question-frequency ranking and the solution filter are not implemented. *Removed in the audit*: "distinct" (the source says "carefully selected, diverse, and representative"), a sentence on "what {partner} is really after" identical to the role sentence of Seed Architecture, a cap of four words on answers (the source uses a Yes/No template), and

our sentence introducing the plan to the action call; the pilot-derived length cap (3000) was reset to 8000.

Zero-shot CoT (Kojima et al., 2022). *Kept.* The first call is *"Q: What should {agent} do in this turn? A: Let's think step by step."* at temperature 0 with the published 128-token reasoning budget. *Truncation.* The budget is binding: in a pilot all 20 reasoning calls reached it, and the truncated JSON output failed to parse, leaving the reasoning empty. As in the original, where truncated reasoning is passed on, we pass the truncated text to the action call; this fallback is enabled for this method only. In the final evaluation all 3244 reasoning calls produced non-empty text. *Lost.* The answer-extraction prompt and stop sequence; reasoning and answer are separate calls. *Removed in the audit*: a checklist of social reasoning steps, including a sentence identical to the role sentence of Seed Architecture; the method was moved from one call to two so that the reasoning is written at all.

ReAct (Yao et al., 2023). *Kept.* The zero-shot ReAct instruction as given in the ReflAct paper (Appendix D.1): *"You should first think about the current condition and plan for your future actions, and then output your action in this turn."*, plus one line of ours adapting it to the environment. *Lost.* No few-shot trajectories; thought and action are two calls rather than one interleaved generation; thoughts are not appended to later context. *Removed in the audit*: "what {agent} still does not know", identical to the role sentence of our probe-then-act planner, and "choose an action whose reply will tell you something".

ReflAct (Kim et al., 2025). *Kept.* Identical to ReAct except for the published instruction *"You should first reflect on the agent's state in relation to the task goal, and then output the action for this turn."*, so ReAct and ReflAct form a minimal pair. *Lost.* The one-shot example; reflection and action are two calls; no official code exists to check against. *Removed in the audit*: "about {partner}", "whether the last observation fits or contradicts that belief" and "what that still leaves unachieved".

GATE (Li et al., 2025a). *Kept.* The question-generation template of its Appendix A.1, verbatim up to domain words: *"Generate the most informative [yes/no question | open-ended question] that, when answered, will reveal the most about … beyond what has already been queried for above"*, with its instructions to address a different aspect, stay bite-sized and avoid jargon. The elicitation policy is one call per turn, as published; the action call speaks the question. The two rows differ in exactly the noun phrase the template leaves open, so they are a true minimal pair. *Lost.* The predictor the paper queries once per test case after elicitation is not ported: our benchmark scores the transcript, not a prediction, and porting it would add a third call.

Plan-and-Solve (Wang et al., 2023). *Kept.* The PS trigger sentence, verbatim—*"Let's first understand the problem and devise a plan to solve the problem. Then, let's carry out the plan and solve the problem step by step."*—inside the same *"Q: [X]. A: [T]"* template as zero-shot CoT, whose environment mapping [X] is byte-identical between the two rows; temperature 0 as published. *Lost.* The PS+ variant, whose two extra sentences concern numerals and intermediate calculations and have no referent in a conversation; self-consistency ($N = 10$ samples), which would multiply the call budget. *Caveat.* PS and zero-shot CoT are not a one-sentence minimal pair: the paper states no generation cap, so its reasoning call runs to our default 8000 tokens against CoT's published 128.

LLM-Deliberation (Abdelnabi et al., 2024). *Kept.* Two blocks of the greedy agent's per-round instructions, verbatim up to domain words: the incentive block (*"You don't put much weight on accommodating … and you would like to maximize … as much as possible"*) and the observation block (*"observe … and see how you can make a move that might maximize"*). The plan written for the agent's own next turn is carried across turns, the only published baseline here with state between turns. *Lost.* Everything the original hangs on numeric scores—the six parties, the issue–option scoring table, the calculator tool, the BATNA threshold and the "deal accepted by at least five parties" criterion—has no counterpart in a two-party conversation and is removed wholesale rather than approximated. *Caveat.* The paper limits each agent to the last six interactions; our observation block, shared by every agent, carries the whole conversation, so this row sees more context than the original specifies.

Table 6: Robustness of the test-set ranking. **Left**: five ways of aggregating the three stored judge votes per episode (τ: Kendall rank correlation of the ten agents with the default three-vote majority). **Right**: splitting the 100 seeds per world into disjoint blocks; each block ranks all ten agents on its own.

Judging rule	τ vs majority
majority of 3 (default)	1.00
first vote only	0.94
unanimous	1.00
any 1 of 3	0.97
fraction of facts found	1.00

Seeds/world	Blocks	Ours > all published	Median τ
10	10	7	0.72
25	4	4	0.86
50	2	2	0.90
100	1	1	1.00

A.5 Evaluation coverage

Each method was evaluated once on the 400 test episodes, and each episode was judged with three votes; all rows share the same Vanilla reference run (§4.1). For every row, all 400 episodes have a transcript and three valid judge votes, no episode is duplicated, and every model call used `deepseek-v3.2`. The evaluations of MediQ, CoT, ReAct, ReflAct and ATD were interrupted once by an expired API credential and resumed; completed episodes are never re-run, so each episode's score comes from exactly one episode. The null action (`none`, with an empty argument), typically chosen after the partner has also stopped, is kept as the method's own behaviour (between 13 and 171 of 3,083–3,977 focal-agent turns for the two-call methods).

B Additional experiments and analyses

Benchmark quality axes. Definitions for Figure 3.

1. **Resolution.** The separation $\mathrm{sd}_{a\in\mathcal{M}}\, S(a;\mathcal{W})/\hat{\varepsilon}$, and the fraction of method pairs whose gap exceeds $\hat{\varepsilon}$. Unlike the spread, both are stated in units of the evaluation noise.
2. **Reliability.** How consistently the benchmark ranks methods: the split-half rank correlation over worlds with the Spearman–Brown correction, and the generalisability coefficient $\sigma^2_{\text{method}}/(\sigma^2_{\text{method}} + \sigma^2_{\text{int}}/|\mathcal{W}| + \sigma^2_{\text{res}}/(|\mathcal{W}||\Sigma|))$ of the two-way method-by-world decomposition.
3. **Item quality.** The fraction of worlds that are *non-discriminative* ($\bar{S}(w) \geq 0.95$ or ≤ 0.05, where $\bar{S}(w)$ averages over $\mathcal{M}$); the discrimination D_w of each world, the gap on w between the stronger and the weaker half of $\mathcal{M}$ ranked by their leave-w-out ability; and the fraction of *discriminative* task families, those whose mean over $\mathcal{M}$ is below 0.85. A family above that line no longer measures the computation it was written to measure.
4. **Construct validity.** Which axis the benchmark separates: the gap between the two host architectures divided by the gap between mechanisms grafted onto one host.

B.1 Behaviour on the test set

The gain is not uniform across world sources. Figure 6b scores WoArc on every evaluation set we have, each against an independent Vanilla run on the same worlds. It gains $+0.1400$ on the test worlds, $+0.0375$ on the architecture-aware worlds and $+0.0391$ on the unevolved benchmark, whose worlds were written before it existed, and falls to -0.1607 on the architecture-independent worlds. An architecture found by search on five worlds from an empty agent can therefore overfit the worlds it was searched on.

The ranking does not hinge on how votes are counted or on which seeds are drawn. Under each of five rules for turning the three judge votes into a verdict, the evolved architectures stay above every published method, and the induced ranking correlates 0.94–1.00 with the default rule (Table 6). When the seeds are split into disjoint blocks, every block of 25 or more seeds per world puts WoArc (from Seed Architecture) above all seven of the published methods that analysis covers, as do 7 of 10 blocks of only 10 seeds.

The evolved architecture is not tuned to a knife-edge. Its one free hyperparameter is how many competing hypotheses the committee keeps. With two or five hypotheses instead of three, every other

prompt line unchanged, it gains $+0.1975$ and $+0.1900$ over Vanilla, against $+0.2400$ for the default of three: an inverted U whose every point still beats the strongest published method ($+0.1750$). The difference lies almost entirely in the commit-first family ($+0.38$, $+0.48$, $+0.33$).

The benchmark result is insensitive to the acceptance thresholds. Sweeping the acceptance band over $[0.25, 0.75]$, $[0.30, 0.70]$ and $[0.35, 0.65]$ and the minimum Oracle gain over 0.15, 0.20 and 0.25, every setting that can be checked against existing scores keeps the evolved benchmark harder for all six +Seed Architecture and +Probe-then-Act variants of published methods. With eight seeds per world, the two narrower bands and all three gain thresholds select identical worlds; only the widest band changes the benchmark, accepting 33 worlds instead of 17.

B.2 Turn budget

Each test transcript is cut after its T-th turn ($T \in \{4, 8, 12, 16\}$) and re-judged with the unchanged three-vote majority protocol; $T = 20$ is the original evaluation. Because the agent prompt shows only "Turn #k" and never the total, the prefix is exactly the episode that an agent with a T-turn budget would have produced. Absolute success rate is given in Table 7.

Table 7: Test-set success rate under a T-turn budget.

T	4	8	12	16	20
WoArc (from Seed Architecture)	0.1250	0.3725	0.6100	0.7300	0.7825
Seed Architecture	0.1675	0.4100	0.6525	0.7575	0.7725
WoArc	0.1000	0.4725	0.6350	0.6775	0.6825
ToMA	0.1375	0.4200	0.5900	0.6550	0.6625
Vanilla	0.0825	0.3750	0.5250	0.5450	0.5425

For the 30-turn budget we re-evaluated Vanilla and Seed Architecture together with WoArc (from Seed Architecture) on seeds 17–66 (200 episodes). On the same episodes the 20-turn scores are 0.7750 (WoArc (from Seed Architecture)) and 0.5500 (Vanilla); at 30 turns they are 0.8100 and 0.5650, so the gap does not close when the budget is extended.

B.3 Failure taxonomy

For every commit-first episode we combine the judge's per-fact verdict with a keyword detector applied to both speakers' lines (recall 0.99 on commit-first episodes credited by the judge). *Elicited, not adopted*: the partner stated the condition and the judge did not credit the focal agent; *refused after the condition*: in addition, a focal-agent line after the partner's statement declines it. All 12 hand-checked refusal lines were genuine refusals. The world's public goal for the focal agent is to create distance from a friend, which is exactly what a goal-anchored plan protects. Table 8 gives the breakdown.

Table 8: Failure taxonomy on commit-first test episodes (share of episodes).

	found	elicited, not adopted	never elicited	refused after condition
WoArc (from Seed Architecture)	0.91	0.09	0.00	0.07
ATD	0.50	0.20	0.30	0.11
Vanilla	0.43	0.56	0.01	0.51
CoT	0.35	0.63	0.02	0.60
ReAct	0.16	0.83	0.01	0.76
ReflAct	0.12	0.88	0.00	0.83

ReAct turns down the condition it unlocked (commit-first world, test set, seed 27). *T12, partner:* "Look, Ethan, I value our friendship a lot. If you need to clear the runway, I can accept that. But there's one condition: the Thursday run stays in the calendar." *T13, ReAct:* "But the Thursday run… I think that has to go on hold too. … even that standing commitment is something I can't reliably keep. I'm sorry." *T14, partner:* "That was my one condition." *T19, ReAct:* "I wish I could give you a different answer, but I can't." *T20, partner:* "If the Thursday run is gone, then… I think we're done here." Judge: not found.

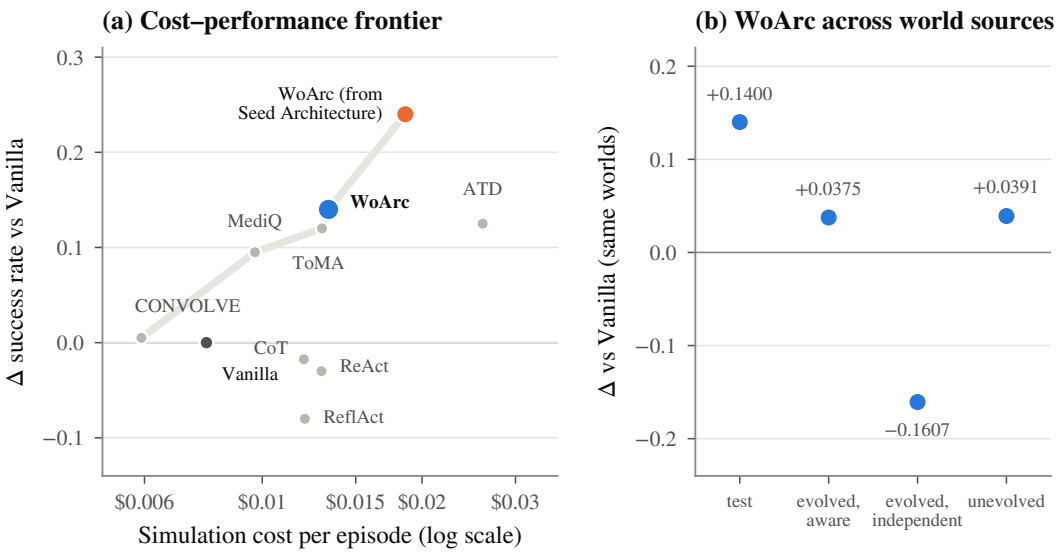

Figure 6: **(a)** Simulation cost per test episode (both speakers, computed from logged per-call token usage) against Δ over Vanilla; the grey line is the Pareto frontier. **(b)** Δ over an independent Vanilla run on each evaluation set for WoArc, the best architecture evolved from Vanilla.

Cost–performance trade-off. Per test episode, WoArc (from Seed Architecture) costs $0.0186 (Figure 6a). The frontier runs CONVOLVE → MediQ → ToMA → WoArc → WoArc (from Seed Architecture), and the most expensive published method is dominated outright: ATD ($0.0260) writes nearly three times as many tokens per episode as WoArc (from Seed Architecture) (11,863 against 4,084) and scores lower.

B.4 Cost accounting

Costs are summed from the per-call cost log over the final evaluation of each agent (both speakers, all calls of the last attempt of each episode; earlier abandoned evaluations of the same agent are excluded). Simulation cost per episode and completion tokens per episode: CONVOLVE $0.0059 / 885; Vanilla $0.0079 / 1,094; MediQ $0.0097 / 1,539; ToMA $0.0130 / 2,244; WoArc $0.0133 / 3,318; WoArc (from Seed Architecture) $0.0186 / 4,084; ATD $0.0260 / 11,863. The whole WoArc pipeline behind Table 2 cost $15.88 for one round of world evolution ($0.93 per accepted world), $44.95 for three architecture generations and $9.71 for screening the slot re-specification candidates.

B.5 Transfer across venues

Each Δ in Figure 6b is paired against a Vanilla evaluation that was run independently of acceptance on the same worlds and seeds (test set: the shared Vanilla reference run; the two evolved generation modes and the unevolved benchmark: the Vanilla evaluation measured in the same batch as WoArc).

B.6 Number of hypotheses

The two variants modify only the hypothesis-update node of the evolved architecture: the rule reads "Keep exactly k hypotheses" and the JSON template lists k entries; the action prompt is byte-identical and the output format is unchanged. In pilot episodes every reasoning call returned exactly k hypotheses. Test-set Δ over Vanilla (two facts / discriminate / commit first / disconfirm): k=2: +0.1975 (−0.31 / +0.28 / +0.38 / +0.44); k=3: +0.2400 (−0.25 / +0.30 / +0.48 / +0.43); k=5: +0.1900 (−0.26 / +0.25 / +0.33 / +0.44).

Table 9: Published methods and our two foundations as slot settings of one agent graph. *State*: what the agent carries about the partner across turns; *weighting*: a score over that state; *move*: a conversational move computed from it. Δ is the test-set difference over Vanilla. CONVOLVE's stopping rule is text in the context and fills no slot in our representation.

Method	State	Weighting	Move	Rationale	Stop	Commit	Mem	Δ
Vanilla	–	–	–	–	–	–	–	0
ReflAct	–	–	–	✓	–	–	–	-0.0800
ReAct	–	–	–	✓	–	–	–	-0.0300
Zero-shot CoT	–	–	–	✓	–	–	–	-0.0175
CONVOLVE	–	–	–	–	(text)	–	–	$+0.0050$
MediQ	–	–	–	✓	✓	–	–	$+0.0950$
ToMA	point	–	–	–	–	–	–	$+0.1200$
ATD	set	–	✓	✓	–	–	–	$+0.1250$
Seed Architecture	set	✓	✓	–	–	–	✓	$+0.2300$

B.7 Random-search control

Under the edit vocabulary used by the search, the space of valid sequences of three successive edits starting from Vanilla contains 100 sequences, which compile to 50 distinct architectures; the search evaluated 5 of them. Six were drawn uniformly from the remaining 45 (with a fixed random seed), all six passed the zero-cost checks, and all eight agents were evaluated in the same batch on the development set (5 worlds $\times$ seeds 1–16). Same-batch Δ over Vanilla (0.5625): WoArc $+0.0750$; random draws -0.0875, -0.0375, -0.0125, 0.0000, $+0.0375$, $+0.0625$. The selection replay reuses the search's own selection code and first reproduces the original Pareto fronts, advancement decisions and populations of all three generations exactly; it then re-applies selection under four rules (per-family Pareto, as in the original search; Pareto with a score tie-break; scalar mean; top-k) on all 80 episodes and on seeds 1–8 or 9–16 only. The random architectures therefore average -0.0062 over Vanilla (from -0.0875 to $+0.0625$), while WoArc reaches $+0.0750$, above all six. The replay also shows why the per-candidate budget matters: with 80 episodes the lineage of WoArc survives every generation under per-family Pareto selection, a scalar mean and a plain top-k; with 40 episodes it is lost at the first or second generation, whichever half of the seeds is used.

B.8 Acceptance-threshold sweep

The sweep re-applies Eq. 2 to the 103 calibrated proposals with bands $[0.25, 0.75]$, $[0.30, 0.70]$, $[0.35, 0.65]$ and oracle-gain floors 0.15, 0.20, 0.25. With eight calibration seeds, Vanilla scores are multiples of 0.125, so the two narrower bands coincide and every floor between 0.15 and 0.25 means "at least 0.25"; the nine settings reduce to two accepted benchmarks (17 and 33 worlds). The criterion "harder for every +Seed Architecture and +Probe-then-Act variant" is checked on every subset for which these variants were scored.

B.9 Published methods as slot settings

The winning methods carry a set of competing readings and a move that acts on it. Compiled into one slot space (Table 9), the two agents with a set-valued state and a move computed from it—ATD and Seed Architecture—take the top two places, while the three agents filling only the rationale slot are exactly the three that never beat Vanilla. Adding a slot does not improve performance; changing what a slot is for does: adding a published mechanism to Seed Architecture moves it by at most $+0.0125$ measured against Seed Architecture in the same batch, whereas the single slot re-specification edit found by search reaches $+0.2400$.

C Case study: one world, one lineage

Figure 7 follows one test world through both halves of the loop. The world belongs to the discriminate family: the partner, Naomi, cares about one of six hats for a hidden reason, and a second reading is equally consistent with everything she says.

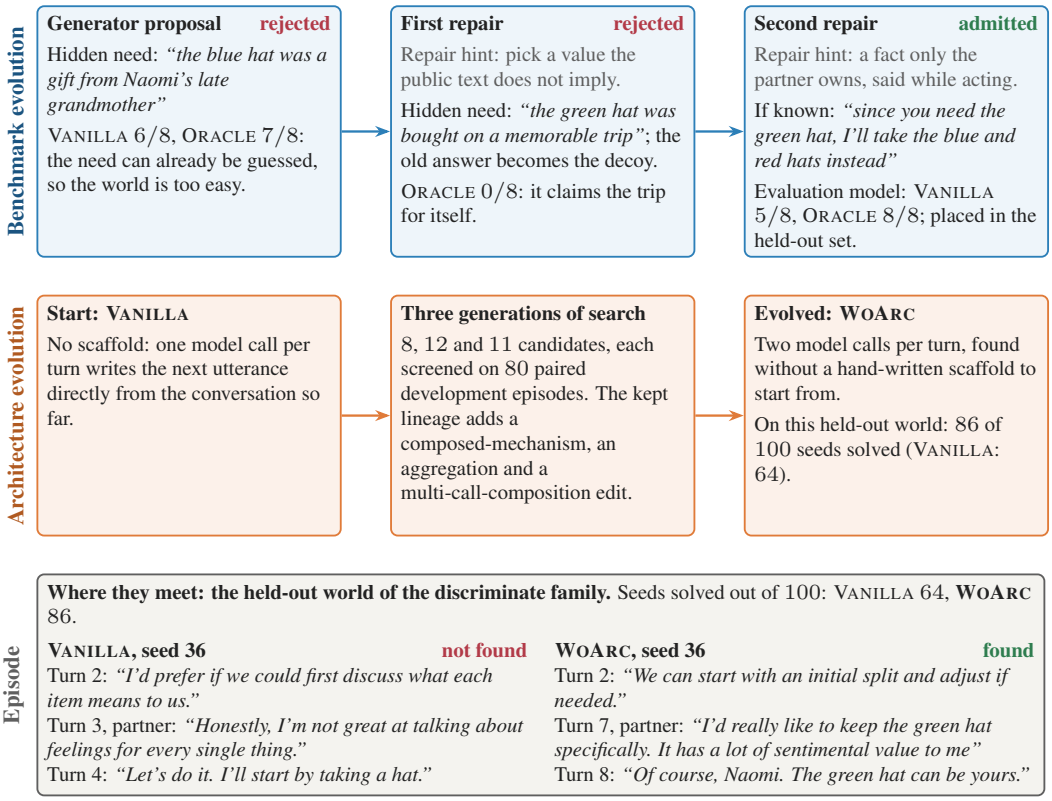

Figure 7: Case study. **Top**: the test world of the discriminate family as the generator evolved it; the need rejected as too easy returns as the decoy of the accepted world. **Middle**: the three generations of search that turn Vanilla into WoArc. **Bottom**: Vanilla and WoArc meet on that world (100 seeds per agent) and in one episode (seed 36, the same world and seed for both agents). Quotes are verbatim.

How the world evolved. The generator's first proposal made the need a gift from Naomi's late grandmother. Vanilla already succeeded on 6 of 8 episodes, so the acceptance criterion rejected the world as TOO EASY: a need that can be guessed tests nothing. The rejection label was returned to the generator as a repair hint—make the correct value something the public text does not imply, and make the common-sense reading the decoy—and the repaired world did exactly that: the special hat became the green one, bought on a trip with a friend who has since died, and the grandmother's blue hat, once the answer, became the trap. The first repair overshot: even Oracle failed, because it claimed the memorable trip for itself. A second hint asked for a fact that only the partner can own and that the agent would naturally say back while acting on it ("since you need the green hat, I'll take the blue and red hats instead"). Re-calibrated on the evaluation model, the world landed at Vanilla 5/8 and Oracle 8/8, inside the acceptance band with a gap to Oracle, and entered the test set. Three rounds turned a guessable preference into a genuine fork whose wrong branch is the most natural guess.

How the architecture evolved. Search starts from Vanilla, which writes each utterance in one model call with no scaffold. Three generations of 8, 12 and 11 candidates, each screened on 80 paired development episodes, keep a lineage that adds in order a composed-mechanism, an aggregation and a multi-call-composition edit (§4.3). The result, WoArc, uses two model calls per turn and was found without any hand-written scaffold to start from.

Where they meet. On this world Vanilla solves 64 of 100 seeds and WoArc 86; WoArc succeeds where Vanilla fails on 32 seeds, and the reverse happens on 10. Seed 36 shows the difference. Vanilla opens by proposing to discuss what each item means to each of them, drops the idea as soon as Naomi demurs, and splits the items by taking turns; the green hat never comes up. WoArc instead proposes an explicit split that it offers to adjust, and in that adjustment Naomi names the green

hat and its sentimental value herself; WoArc gives it to her at once. The world was evolved until guessing lands on the decoy; the architecture was evolved until the conversation leaves the partner room to state the need.

World versions. All three versions share the public scenario (two players split 4 books, 6 hats and 2 balls; partner Naomi Fletcher, focal agent Sophia Brown) and the partner rule of the discriminate family: the partner answers in a way that separates the two readings only when a question would be answered differently under them. Table 10 lists the fields that differ.

Table 10: Case study: fields of the proposal and its two repairs.

Field	Proposal	Repair 1	Repair 2 (test)
correct value	the blue hat was a gift from {partner}'s late grandmother	the green hat was bought on a memorable trip	same as repair 1
decoy	the red hat was a gift from {partner}'s sibling	the blue hat was a gift from {partner}'s late grandmother	same as repair 1
action if known	leave the blue hat to Naomi Fletcher	leave the green hat to Naomi Fletcher	leave the green hat to Naomi Fletcher and say: 'since you need the green hat, I'll take the blue and red hats instead.'
calibration	Vanilla 6/8, Oracle 7/8	Vanilla 0/8, Oracle 0/8	Vanilla 5/8, Oracle 8/8 (evaluation model)
verdict	too easy	rejected (oracle cannot succeed)	accepted

In repair 1 the oracle agent volunteered the sentiment as its own (seed 2, T6: "I do have a soft spot for the green one, though—it reminds me of a memorable trip."), which is what the second repair hint targets: "make sure only the partner could truthfully claim it, so the focal cannot satisfy it by talking about itself."

Re-specification candidates. Δ against Seed Architecture measured in the same batch on the 64-episode development set (discriminate family and overall) are listed in Table 11.

Table 11: Case study: re-specification candidates and their paired Δ against Seed Architecture on the development set.

New slot text	Author	Discriminate	All
Ask a question whose answer would confirm one of the top two hypotheses and contradict the other.	sighted	$+0.125$	$+0.000$
Ask a question whose answer would be different depending on which of the top two hypotheses about {other}'s true goal is correct.	sighted	0.000	-0.031
Ask {other} a question that forces them to reveal which of the two most likely hidden facts they actually know.	sighted	-0.313	-0.141
Ask a question or make a statement that would elicit a different response depending on which of the two leading hypotheses about {other}'s true goal is correct.	blind	0.000	-0.063
a move that tests whether {other} will accept a final offer or keep negotiating	blind	-0.438	-0.203

Episode-level counts on the test discriminate world (success of Vanilla, Seed Architecture and WoArc (from Seed Architecture) per seed): WoArc (from Seed Architecture) succeeds where Vanilla fails on 33 seeds and the reverse holds on 3; it succeeds where Seed Architecture fails on 10 seeds and the reverse holds on 5. Seed 36 is the only seed with the pattern (Vanilla fails, Seed Architecture fails, WoArc (from Seed Architecture) succeeds) on which all three judge votes agree for all three agents.

D STOPPING RULE: PROOF, CONSTANTS AND REPLAY

Stopping loss. Stopping trades two losses: the budget spent on generations whose best candidate is indistinguishable from noise, and the gain forfeited by stopping while an improvement of size g is still reachable. With per-generation cost c_t,

$$\mathcal{L}(\tau) \;=\; \sum_{t \le \tau} c_t \left(1 - \pi_t\right) \;+\; \lambda \mathbb{1}\big[\exists\, t > \tau :\ \text{generation } t \text{ contains a gain} \ge g\big], \tag{8}$$

Algorithm 1 One generation of WoArc

Require: architecture population $\mathcal{P}_t$, benchmark $\mathcal{W}_t^\star$, unused scenarios $\mathcal{U}$, edit vocabulary $\mathcal{S}$, generator G_ϕ

// World evolution
for each scenario $s \in \mathcal{U}$ assigned to generation mode $r \in \{\text{aware}, \text{indep}\}$ and family f **do**
  $w \leftarrow G_\phi(\cdot \mid p_B(s, f) \oplus \mathbb{1}[r{=}\text{cond}]\, m)$; record $(r, m, \text{generator model}, \text{prompt hash})$
  if w passes the zero-cost validity checks **then**
    estimate $S(Vanilla; w)$, $S(Oracle; w)$ with $n = 8$ seeds
    if Eq. 2 holds **then**
      $\mathcal{W}_{t+1}^\star \leftarrow \mathcal{W}_t^\star \cup \{w\}$
    else
      record the rejection label
    end if
  end if
end for
// Architecture evolution
$\mathcal{B} \leftarrow \emptyset$
for each parent $p \in \mathcal{P}_t$ and mechanism family **do**
  build candidate edit b by chaining edit types from $\mathcal{S}$ on p (slot re-specification text from the author model)
  if causal chain (i)–(iv) certified by the zero-cost operativity checks **then**
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{b\}$
  end if
end for
evaluate every $b \in \mathcal{B}$ and its parent on shared episodes; compute $\boldsymbol{\delta}(b \mid p)$ (Eq. 4)
$\mathcal{F}_t \leftarrow$ Pareto front of $\{\boldsymbol{\delta}(b)\}$
$\mathcal{P}_{t+1} \leftarrow \mathcal{P}_t \cup \{b \in \mathcal{F}_t : \max_f \delta_f(b) > 0\}$
// Stopping rule (§3.3)
compute π_t and H_t (Eq. 7); update record R_t and stall count k_t
if $H_t \le \varepsilon_{m_t}(n_t)$ **then**
  next generation evolves worlds only
else if the stopping condition of Eq. 6 holds **then**
  stop architecture evolution from this foundation
end if
return $\mathcal{P}_{t+1}, \mathcal{W}_{t+1}^\star$

where λ prices a missed improvement; Eqs. 5–6 bound both terms without knowing the distribution of gains. Part (i) of Proposition 1 follows from Eq. 5: the record is the maximum of earlier estimates and is therefore biased upwards, which can only lower the chance that a candidate with no true gain clears it. Part (ii) follows because stopping requires a run of misses on independent evaluations whose probabilities, each at most $1 - \beta_s(g)$ when the record is treated as fixed, multiply to at most δ.

D.1 Proof of Proposition 1

(i) Suppose no candidate in generation t improves on the architecture that holds the record. The record R_{t-1} is the maximum of earlier noisy estimates, so $R_{t-1} \ge S(a_{\text{rec}}) + \xi_0$ in distribution for an independent $\xi_0 \sim \mathcal{N}(0, \sigma_n^2)$. Hence $\Pr[\pi_t = 1] \le \Pr[\max_{i \le m}(S(a_i) + \xi_i) - S(a_{\text{rec}}) - \xi_0 > \varepsilon_m(n)] \le \Pr[\max_{i \le m} \xi_i - \xi_0 > \varepsilon_m(n)] = \alpha$, using $S(a_i) \le S(a_{\text{rec}})$ and the definition of $\varepsilon_m(n)$ in Eq. 5.

(ii) Suppose each generation $s > t_0$ contains a candidate a_s^+ with $S(a_s^+) \ge R + g$, where R is the record treated as fixed. Then $\Pr[\pi_s = 0] \le \Pr[\xi_{a_s^+} - \xi_0 \le \varepsilon_{m_s}(n_s) - g] = 1 - \beta_s(g)$, because the difference of two independent noise terms has variance $\hat{q}/n_s$. Evaluations of different generations use fresh episodes and are independent, so the probability that a stall run reaches the stopping condition of Eq. 6 is at most $\prod_s (1 - \beta_s(g)) \le \delta$. □

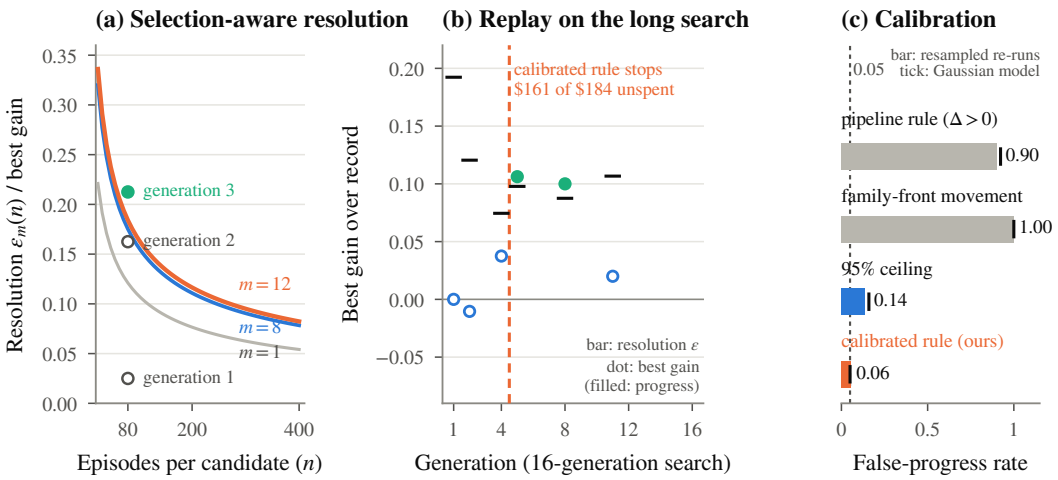

Figure 8: **(a)** Selection-aware resolution $\varepsilon_m(n)$ (Eq. 5) for $m \in \{1, 8, 12\}$ candidates, with the best gain of each generation evolved from Vanilla (filled: registered as progress). **(b)** Retrospective replay of the rule on an earlier 16-generation architecture search: bars give the resolution of each search generation, dots its best gain over the record; the rule stops after generation 4. **(c)** Rate at which a generation containing no real improvement is declared progress, for $n = 80$ episodes and $m = 12$ candidates (bars: resampling two independent runs of the same architecture; ticks: the Gaussian model).

D.2 Resolution table

$\varepsilon_m(n)$ at $\hat{q} = 0.4333$ and $\alpha = 0.05$ (closed form; a Monte-Carlo check agrees within 0.0003) is given in Table 12.

Table 12: Selection-aware resolution $\varepsilon_m(n)$ at $\hat{q} = 0.4333$ and $\alpha = 0.05$.

n	$m = 1$	$m = 4$	$m = 8$	$m = 12$
64	0.1353	0.1778	0.1959	0.2058
80	0.1211	0.1590	0.1753	0.1841

D.3 Replay on an earlier search

We evaluate the stopping rule of §3.3 on every architecture search the project has run, at no additional cost: all quantities it needs are already recorded. The evaluation noise is measured from repeated Vanilla runs ($\hat{q} = 0.4333$ on the development set used for evolution from Vanilla).

Existing progress tests mistake noise for progress. Figure 8c compares four ways of declaring that a generation has improved, applied to generations in which no candidate truly improves. Accepting any positive gain, as most evolutionary pipelines do, declares progress in 0.90 of such generations; movement of the per-family Pareto front does so in all of them; even a conventional 95% threshold that ignores selection among 12 candidates does so in 0.14. The selection-aware threshold of Eq. 5 holds the rate at 0.06, matching its design level $\alpha = 0.05$. On the 22 exact comparisons between two independent runs of the same architecture, it declares progress 0 times, against 11 for the positive-gain test and 18 for front movement.

The rule keeps what evolution found. Evolving from Vanilla, the first two generations' best candidates (+0.025 and +0.1625) stay within their resolution (0.1753 and 0.1841), so neither counts as progress; the accumulated evidence against a gain of $g = 0.14$ after these two generations is still weak (miss probability $0.496 > \delta$), so the rule continues, and generation 3 clears its resolution (+0.2125 against 0.1823). The architecture that becomes WoArc is therefore kept for every $\delta \in \{0.05, 0.10, 0.20\}$ (Appendix D).

The rule stops what could not succeed. An earlier automated search ran for 16 generations on an earlier version of the benchmark. Replaying the rule on its search generations (Figure 8b), no candidate clears its resolution in generations 1, 2 and 4, whose evaluations are precise enough that the miss probability for $g = 0.14$ falls to 0.018; the rule stops after generation 4. Of the campaign's \$184, \$161 was spent after that point; it included two apparent improvements (generations 5 and 8), and both failed later confirmation: one did not beat a structure-matched control and the other lost on test worlds. The same conclusion holds for every $\delta \in \{0.05, 0.10, 0.20\}$ at $g = 0.14$.

The gap to Oracle tells the two kinds of stall apart. On the development worlds used for evolution from Vanilla, the gap to Oracle of Eq. 7 stays above the resolution throughout (H_t/ε between 1.51 and 2.65), so the rule attributes a stall there to the edit vocabulary rather than to the worlds, and prescribes continuing architecture evolution rather than a new round of world evolution—the decision the search took.

D.4 Replay: per-generation detail

Evolution from Vanilla ($n = 80$ development episodes, $\hat{q} = 0.4333$): best gain over the record and resolution per generation are $+0.025$ / 0.1753 ($m = 8$), $+0.1625$ / 0.1841 ($m = 12$) and $+0.2125$ / 0.1823 ($m = 11$). With $g = 0.14$ the per-generation detection probabilities of the first two generations are 0.316 and 0.275, leaving a miss probability of 0.496 before generation 3 registers progress.

Earlier 16-generation search ($\hat{q} = 0.3280$, pooled from 9 pairs of consecutive Vanilla runs, 1,320 episodes): the search generations were 1 ($n = 24$, $m = 1$), 2 ($n = 96$, $m = 3$), 4 ($n = 160$, $m = 1$), 5 ($n = 160$, $m = 4$), 8 ($n = 200$, $m = 4$) and 11 ($n = 200$, $m = 2$), with best gains over the record of 0.000, -0.010, $+0.038$, $+0.106$, $+0.100$ and $+0.020$ against resolutions 0.1923, 0.1205, 0.0745, 0.0978, 0.0875 and 0.1067. At $g = 0.14$ the detection probabilities of generations 1, 2 and 4 are 0.327, 0.631 and 0.926, and the miss product reaches 0.018 after generation 4.

Sensitivity. Stopping decisions for the two target gains we consider meaningful at this budget are listed in Table 13.

Table 13: Sensitivity of the stopping decisions to the target gain g and the miss probability δ.

g	δ	Evolution from Vanilla	16-generation search	Unspent
0.14	0.05	continues, WoArc kept	stops after generation 4	\$161
0.14	0.10	continues, WoArc kept	stops after generation 4	\$161
0.14	0.20	continues, WoArc kept	stops after generation 4	\$161
0.20	0.05	continues, WoArc kept	stops after generation 2	\$175
0.20	0.10	continues, WoArc kept	stops after generation 2	\$175

D.5 Calibration against pure noise

For each generation we simulate m candidates and a parent with no true difference, either under the Gaussian model with the measured $\hat{q}$ or by resampling episode-level verdicts of two independent runs of the same architecture, and record how often each test declares progress. At $n = 80$ and $m = 12$ the rates are 0.923 / 0.904 (any positive gain), 1.000 / 1.000 (per-family front movement), 0.159 / 0.137 (a 95% threshold without selection correction) and 0.050 / 0.056 (Eq. 5), for the Gaussian model and resampling respectively. Across all resampled settings the calibrated rate lies between 0.042 and 0.057.