ARGGYM: A PROCEDURAL, ENGINE-VERIFIED BENCHMARK FOR STRUCTURED DEFEASIBLE REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent progress in large language model reasoning has been driven by benchmarks and reinforcement learning environments with automatically verifiable rewards, particularly in mathematics, code, and formal logic. These settings make model accuracy easier to evaluate and optimize, but it remains unclear how far success under fixed problem specifications and stable evaluation criteria transfers to reasoning outside such domains. Real-world reasoning often proceeds under incomplete and revisable information: conclusions may be supported provisionally, defeated by counter-evidence, reinstated by further arguments, or revised when stronger reasons become available. Reasoning of this kind is generally referred to as *defeasible reasoning*. We introduce ArgGYM, a procedural benchmark and RLVR-compatible training environment for structured defeasible reasoning. ArgGYM decomposes this reasoning into twelve tasks and grounds task-specific scoring in a symbolic argumentation engine that computes the formal states used to evaluate model outputs. It includes a frozen benchmark of 1,440 verified instances across fifteen curriculum configurations, two argument preference orderings (weakest-link and last-link), and two set orderings (elitist and democratic), while the same generators and verifiers can produce fresh instances for evaluation that reduces dependence on static test sets and for verifiable-reward training. On the frozen benchmark, frontier and open-weight models show sharply different reasoning profiles: they can recover substantial parts of structured answers without solving the complete task, and performance declines in later curriculum configurations with longer dependencies and more interacting structures. We release the benchmark, generators, and verifiers for reproducible evaluation and RLVR training.

1 INTRODUCTION

Advances in large language models and reasoning-focused models have been driven in part by reinforcement learning with verifiable rewards (RLVR), especially in mathematics, code, and formal logic, where solution accuracy can be checked automatically (Shao et al., 2024; Guo et al., 2025; Xie et al., 2025b). These domains are important tests of reasoning because they require precise inference and often produce outcomes that can be checked against explicit criteria. Many real-world reasoning problems, however, do not have fixed specifications or stable criteria in the same way. A lawyer may revise an earlier judgment when new evidence is introduced or when competing reasons must be weighed against one another; a physician may have to reach a decision from incomplete information while remaining prepared to revise it as new evidence becomes available. Reasoning of this kind is commonly treated as defeasible: conclusions are warranted given the available information, but may later be withdrawn when stronger or conflicting reasons arise. It is therefore non-monotonic, in the sense that adding new information can invalidate a conclusion that was previously supported. Defeasible and non-monotonic reasoning have long been studied in artificial intelligence, cognitive science, legal reasoning, and argumentation as models of reasoning under incomplete information, conflict, exceptions, and revision (Reiter, 1980; Pollock, 1987; Prakken & Sartor, 1997; Prakken, 2010). If language models are to operate reliably in domains where information is incomplete and judgments remain open to revision, the ability to reason defeasibly is likely to be an important part

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.

of their broader reasoning competence. Evaluating this ability is therefore necessary for understanding how well current models can handle reasoning under incomplete, conflicting, and revisable information.

Recent efforts have begun to evaluate the defeasible and non-monotonic reasoning capabilities of LLMs (Xiu et al., 2022; Kazemi et al., 2023; Parmar et al., 2024; Wilie et al., 2024; Allaway & McKeown, 2025; Li et al., 2025). These studies report weaknesses in generalization, belief revision, and reasoning under conflicting or revisable information. More recent work has also developed procedural or formally verified benchmarks and generation frameworks for particular defeasible reasoning problems (Cooper & Velasquez, 2026; Leiva et al., 2026; Sadhu et al., 2026). Yet the evaluation and training infrastructure for defeasible reasoning remains much less developed than in mathematics, code, and deductive logic, where procedural generation and exact verification can be used both to benchmark models and to provide training rewards.

Taken together, these developments leave open a complementary gap: a unified executable environment in which multiple structured defeasible-reasoning tasks are instantiated over the same formal substrate, separately verified under task-specific contracts, and varied systematically in their structural and preference-sensitive demands. Such an environment should support both a frozen benchmark for reproducible model comparison and fresh procedural generation using the same verification machinery. This combination is useful for evaluation because it exposes operation-specific capability profiles within a common formal system, while for reinforcement learning it enables fresh verified training instances without drawing from the frozen evaluation suite. Procedural generation can reduce dependence on static evaluation sets and their associated contamination risks (White et al., 2025; Chen et al., 2025b; Xie et al., 2025a), while mechanically verifiable outcomes can serve directly as rewards for reinforcement learning (Stojanovski et al., 2025).

Structured argumentation provides a natural formal basis for this setting. Dung (1995) established a fundamental connection between argumentation and non-monotonic reasoning, showing how forms of non-monotonic inference can be understood in terms of arguments, attacks, and acceptability. Structured argumentation frameworks such as ASPIC+ additionally make the internal construction of arguments explicit: arguments are built from premises and inference rules, conflicts give rise to attacks, preferences determine whether preference-sensitive attacks succeed as defeats, and argumentation semantics determine which arguments, and consequently which conclusions, are accepted (Prakken, 2010; Modgil & Prakken, 2014). Because the resulting argumentative states can be computed mechanically, structured argumentation supports both controlled decomposition of reasoning capabilities and executable verification of model outputs.

ArgGYM operationalizes this connection as an ASPIC+-based procedural benchmark and RLVR-compatible environment. Its twelve tasks cover formalization and support recovery, defeat diagnosis, semantic evaluation, theory revision, preference reasoning, and constructive intervention. Generated theories vary in structural composition, argument preference ordering, and set ordering, while a symbolic argumentation engine computes or validates the formal states required by the task-specific scorers. For reproducible comparison, ArgGYM includes a frozen benchmark of 1,440 verified instances spanning all twelve tasks, fifteen curriculum configurations, two argument preference orderings (weakest-link and last-link), and two set orderings (elitist and democratic). The same generators and verifiers can also produce fresh verified instances.

Across 18 distinct model variants spanning frontier and open-weight models, performance varies markedly across reasoning operations: models with similar aggregate accuracy can exhibit different task profiles, and success on formalization or support recovery does not imply reliable semantic evaluation, defeat analysis, revision, or intervention. These results motivate evaluating structured defeasible reasoning at the level of its component operations rather than as a single undifferentiated capability.

Our contributions are:

- We introduce ArgGYM, a twelve-task procedural environment for structured defeasible reasoning based on ASPIC+.
- We provide a frozen 1,440-instance benchmark together with procedural generators that produce fresh, engine-verified instances across controlled curriculum configurations, argument preference orderings, and set orderings.

- We expose the same task-specific generators and symbolic verifiers through an RLVR-compatible interface for mechanically rewarded training.

2 Related Work

Defeasible reasoning. Several benchmarks test language models in settings where conclusions can change as information is added or conflicts arise. δ-NLI asks whether additional evidence strengthens or weakens a natural-language inference (Rudinger et al., 2020); LogicNMR isolates non-monotonic reasoning through explicit default rules and iterative updates (Xiu et al., 2022); BoardgameQA introduces contradictory information resolved through preferences over information sources (Kazemi et al., 2023); and Belief-R tests whether models revise earlier conclusions when new evidence becomes available (Wilie et al., 2024). More formal approaches evaluate rule-based defeasible reasoning directly. These include Defeasible Logic reasoning patterns (Tachmazidis et al., 2024), formal defeasible reasoning over synthetic DeLP programs with varying reasoning depth (Li et al., 2025), and defeasible property inheritance in DEFREASING (Allaway & McKeown, 2025). These benchmarks establish defeasible reasoning as a distinct LLM evaluation problem, but generally focus on particular inference problems or formalisms rather than decomposing structured argumentation into a broader set of separately verifiable operations.

Computational argumentation. Recent work has examined the relationship between LLMs and argumentation from two directions. In one, argumentation itself is treated as a model capability: evaluations cover natural-language argument mining and generation (Chen et al., 2024), ArgBench unifies 33 datasets across 46 computational argumentation tasks (Ajjour et al., 2026), and work on abstract argumentation evaluates and trains language models to compute argument acceptability over generated argumentation frameworks (Li et al., 2026). In the other direction, argumentation is used as formal machinery for improving, verifying, or revising LLM outputs. MQArgEng introduces formal argumentation semantics into an LLM pipeline (Castagna et al., 2024); ArgLLMs construct argumentation frameworks that support explainable and contestable claim verification (Freedman et al., 2025); LLM-ASPIC$^+$ delegates conflict resolution to an ASPIC$^+$ component after LLM-based grounding and reasoning (Fang et al., 2026); and ARGUS represents LLM self-explanations as Dung-style argumentation frameworks and computes minimum-cost repairs when new evidence changes the required acceptability status (Xiao & Li, 2026).

Procedural and verifiable reasoning. Synthetic reasoning benchmarks established that formally generated theories can support controlled evaluation of model reasoning. RuleTaker generates natural-language facts and rules for multi-step inference (Clark et al., 2020), while ProofWriter extends this setting with explicit proofs and abductive reasoning (Tafjord et al., 2021). ProverGen combines scalable data generation with symbolic prover verification to construct first-order-logic reasoning problems and their intermediate reasoning steps (Qi et al., 2025). More recently, LogicSkills separates formal reasoning into symbolization, countermodel construction, and validity assessment, with instances checked using an SMT solver (Rabern et al., 2026). Reasoning Gym and Enigmata scale this pattern into multi-task procedural environments with controllable difficulty and programmatic verification suitable for RLVR (Stojanovski et al., 2025; Chen et al., 2025a). Related developments are now appearing in defeasible reasoning itself. DeFAb procedurally constructs defeasible-abduction problems with an exact verifier that can serve as a training reward (Cooper & Velasquez, 2026); DeReLab generates formally verified belief-updating problems across default and inheritance reasoning (Sadhu et al., 2026); and DPG generates synthetic DeLP knowledge bases with controllable structural properties for benchmarking structured-argumentation systems (Leiva et al., 2026).

Positioning. ArgGYM is complementary to these lines of work rather than a replacement for them. ArgBench provides broad coverage across heterogeneous computational argumentation datasets, while DeFAb and DeReLab develop verifiable or generative benchmarks for particular defeasible-reasoning problems, and DPG provides controlled procedural generation of structured DeLP knowledge bases. Broader environments such as Reasoning Gym demonstrate the value of procedural generation and exact verification for multi-task reasoning and RLVR. ArgGYM's contribution is to bring these properties together within a single executable structured-argumentation environment: its benchmark tasks share an ASPIC+ substrate and common engine semantics, span both analysis and

constructive intervention, include a matched intervention-language ablation, and use task-specific verification contracts that apply both to the frozen benchmark and to freshly generated instances. This common substrate makes it possible to compare operation-specific and structural failure modes without changing the underlying argumentation formalism, while retaining mechanically verifiable outputs suitable for future training.

3 ARGGYM

This section describes ArgGYM's formal substrate, procedural generation process, task suite, and frozen evaluation benchmark.

3.1 FORMAL SETTING AND VERIFICATION

An ASPIC+ argumentation theory consists of a knowledge base, strict and defeasible inference rules, conflict relations, and preferences (Modgil & Prakken, 2014). Arguments are constructed recursively from premises by applying strict or defeasible rules. Conflicts give rise to three standard attack types: *undermining* targets an ordinary premise, *rebutting* targets a conclusion reached defeasibly, and *undercutting* targets the applicability of a defeasible inference. Preferences determine whether preference-sensitive attacks succeed as defeats, and argumentation semantics determine which arguments, and consequently which claims, are accepted.

ArgGYM uses a pinned PyArg backend for executable ASPIC+ computation (Odekerken et al., 2023). ArgGYM supplies the task-specific procedural layer: it instantiates generated theories, applies the selected argument preference ordering and set ordering, and issues the formal queries required by each task. The symbolic engine is used for generation and verification and is not available to the model during inference; it validates candidate instances and, for constructive tasks, executes submitted interventions and recomputes the resulting argumentative state. Constructive outputs can therefore be evaluated by their formal consequences, while non-constructive outputs are scored against task-specific structured targets that are either engine-computed or generator-constructed and engine-validated.

ArgGYM crosses two argument preference orderings, *last-link* and *weakest-link*, with two set orderings, *elitist* and *democratic*, yielding four preference configurations. All tasks except `semantics_query` use grounded semantics, which has a unique extension. The `semantics_query` task additionally requires reasoning under alternative argumentation semantics and acceptance modes. Full definitions and implementation details are given in Appendix A.1.

3.2 PROCEDURAL GENERATION

Each instance is generated from a task $\tau \in \{1, \ldots, 12\}$, curriculum configuration $\ell \in \{1, \ldots, 15\}$, argument preference ordering $p \in \{\text{last-link}, \text{weakest-link}\}$, set ordering $c \in \{\text{elitist}, \text{democratic}\}$, and random seed s:

$$x = G(\tau, \ell, p, c, s). \tag{1}$$

Generation follows a *generate–verify* procedure. A task-specific generator first constructs a candidate theory and task instance. Structural gates then check that the candidate exhibits the reasoning phenomenon required by the task, after which the symbolic engine computes the relevant argumentative state. Candidates that fail either the structural or formal checks are rejected. Accepted instances therefore carry verification targets that are either computed directly by the symbolic engine or constructed by the task generator and validated against engine-computed states and task-specific structural conditions.

The fifteen curriculum configurations vary the structural composition of the generated problems. Depending on the task, they control factors such as derivation depth, branching, interacting attackers and defenders, mixtures of strict and defeasible inference, distractors, preference-sensitive conflicts, support-structure junctions, and multi-goal interactions. These configurations represent increasing structural richness rather than a guaranteed monotonic ordering of empirical difficulty. Together with the argument preference ordering, set ordering, and random seed, they define a controlled space from which both frozen and fresh task instances can be generated.

Table 1: The twelve ArgGYM tasks. The first six require structured analysis or formalization outputs, while the final six require constructive interventions whose consequences are executed and verified through the task-specific scoring pipeline.

Task	Objective
Representation and support	
`formalization`	Construct an executable ASPIC+ theory from a controlled natural-language description whose induced argumentative state is behaviorally equivalent to the intended theory.
`claim_chain`	Recover the complete support derivation for a justified target claim in an ordering consistent with its derivational dependencies.
Diagnosis, semantics, and revision	
`defeat_diagnosis`	Identify where support for a non-justified target fails and the relevant defeaters responsible for that failure.
`status_query`	Determine the statuses of queried claims under grounded semantics.
`semantics_query`	Determine claim acceptance under a specified argumentation semantics.
`perturbation`	Identify which claims of the original theory change status after a specified update and predict the new status of each changed claim.
Constructive intervention	
`preference_construction`	Add preferences that produce a requested configuration of claim statuses.
`counter_argument`	Construct an intervention that justifies the opposing claim without adding strict rules.
`counter_argument_strict`	Strict-rule-enabled condition paired with `counter_argument` on the same underlying theory and semantic objective.
`attack`	Modify the theory so that a justified target becomes overruled.
`defence`	Modify the theory so that an attacked target becomes justified.
`attack_defense`	Modify the theory so that one target becomes overruled while another becomes justified.

3.3 Task Suite

ArgGYM operationalizes structured defeasible reasoning through twelve tasks over the same ASPIC+ formal substrate. The suite covers representation and support recovery, diagnosis and semantic evaluation, revision prediction, and constructive intervention. This decomposition allows model performance to be localized to particular reasoning operations. Table 1 summarizes the tasks.

The six constructive tasks are scored by executing the retained legal intervention and checking the resulting argumentative state. Different interventions can therefore succeed if they satisfy the task-specific requirements; exact reproduction of a stored reference intervention is not required. The remaining six tasks produce structured outputs, including executable formalizations, support derivations, diagnostic records, and claim-status maps.

3.4 Frozen Benchmark

To evaluate models, we freeze a 1,440-instance benchmark sampled from the ArgGYM generation space:

$$12 \times 15 \times 2 \times 2 \times 2 = 1{,}440. \tag{2}$$

The benchmark contains twelve tasks, fifteen curriculum configurations, two argument preference orderings, two set orderings, and two accepted instances for each resulting configuration. These instances remain fixed across all model evaluations. The procedural generators remain separate from the frozen suite and can produce additional verified instances when fresh problems are required.

The same task-specific scoring contracts are used for frozen and newly generated instances. On fresh instances, their verifier outputs can also serve as mechanically computed rewards, making ArgGYM RLVR-compatible without a separate learned reward model.

4 EXPERIMENTAL SETUP

4.1 MODELS AND INFERENCE CONFIGURATION

We evaluate 18 distinct language-model variants from five model families, yielding 42 model–reasoning configurations. The evaluation panel spans Gemma, GPT-OSS, Qwen, DeepSeek, and Gemini systems and covers a broad range of model scales and inference-time reasoning settings. Gemini models are accessed through the provider API, while DeepSeek-v4.1-Flash is accessed through a hosted API. The remaining open-weight Gemma, GPT-OSS, and Qwen models are served on high-performance computing infrastructure using NVIDIA H200 GPUs.

Where supported, we evaluate multiple inference-time reasoning configurations for the same underlying model. Gemma and Qwen3.5 models are evaluated with thinking enabled and disabled. GPT-OSS models are evaluated at low, medium, and high reasoning effort, and Qwen3.8-27B is evaluated at low, medium, and xhigh reasoning effort. DeepSeek-v4.1-Flash is evaluated at high reasoning effort, while Gemini models are evaluated at low, medium, and high reasoning effort.

Every configuration is evaluated on the same frozen 1,440-instance benchmark using the same taskset, prompt version, response template, and scoring version. Inference settings are fixed within each model–reasoning configuration, while model-specific output limits, sampling parameters, context windows, numerical precision, and serving configurations may differ across systems. Complete model identifiers and recorded inference configurations are reported in Appendix D.1.

4.2 EVALUATION METRICS AND SUCCESS CRITERIA

We report *complete-task success* and a *task-native score*. Complete-task success is our primary benchmark metric and indicates whether the full task-specific output contract is satisfied. For structured prediction tasks, this requires exact recovery of the required structured output. For constructive tasks, the submitted intervention must satisfy all requested postconditions after execution while leaving the resulting theory consistent. The `formalization` task is evaluated behaviorally: complete success requires the submitted executable theory to reproduce the reference theory's engine-derived statuses over all literals named by the reference theory, without requiring exact reconstruction of its directives.

Each task also returns a native score in $[0, 1]$ that captures task-specific partial correctness. Depending on the task, this reflects partial recovery of structured outputs, dependency-order correctness, diagnostic accuracy, behavioral and structural agreement, semantic progress, or intervention economy. Because these quantities have different semantics across tasks, native scores are interpreted only within the corresponding task and are never averaged into a benchmark-wide score. Full task-specific scoring definitions are given in Appendix B.

5 RESULTS

5.1 OVERALL PERFORMANCE AND CAPABILITY PROFILES

ArgGYM separates the evaluated systems over a wide range of complete-task success, from near-zero performance in the smallest configurations to 82.4% for Gemini 3.8 Flash at high reasoning effort (Table 2). Additional inference-time reasoning generally helps: Gemma-4-31B rises from 14.4% to 27.3% with thinking enabled, GPT-OSS-120B from 12.4% at low effort to 29.0% at high effort, and Qwen3.8-27B from 28.4% to 39.2% at xhigh effort. The effect is nevertheless model dependent rather than uniformly monotonic; Gemini 2.5 Pro, for example, peaks at medium rather than high effort.

Aggregate performance also does not correspond to a uniform capability profile. Even Gemini 3.8 Flash at high effort, which attains the highest benchmark-wide success, is near ceiling on several tasks yet reaches only 57.5% complete success on `defeat_diagnosis` and 60.0% on the strict-rule-enabled counter-argument condition. Lower-scoring systems likewise show different mixtures of strengths and weaknesses rather than uniformly scaled versions of the same behavior.

Four findings summarize the main empirical patterns:

Table 2: **Overall ArgGYM performance.** Complete-task success (%) across reasoning configurations.

Model	Complete-task success (%)
Gemma	
Gemma 4 E2B	off **1.1** → on **1.3**
Gemma 4 E4B	off **3.5** → on **3.8**
Gemma 4 26B-A4B	off **9.0** → on **14.1**
Gemma 4 31B	off **14.4** → on **27.3**
GPT-OSS	
GPT-OSS 20B	low **6.2** / med. **16.0** / high **21.2**
GPT-OSS 120B	low **12.4** / med. **21.5** / high **29.0**
Qwen	
Qwen3.5 0.8B	off **0.0** → on **0.0**
Qwen3.5 2B	off **0.3** → on **1.0**
Qwen3.5 4B	off **12.4** → on **22.4**
Qwen3.5 9B	off **17.8** → on **26.1**
Qwen3.5 35B-A3B	off **21.7** → on **31.7**
Qwen3.5 27B	off **31.5** → on **39.3**
Qwen3.8 27B	low **28.4** / med. **29.4** / xhigh **39.2**
DeepSeek	
DeepSeek V4.1 Flash	high **45.0**
Gemini	
Gemini 2.5 Pro	low **16.6** / med. **34.5** / high **33.2**
Gemini 3.5 Flash Lite	low **18.2** / med. **29.3** / high **35.1**
Gemini 3.1 Pro Preview	low **48.3** / med. **54.9** / high **62.9**
Gemini 3.8 Flash	low **61.5** / med. **77.8** / high **82.4**

Partial argumentative competence does not reliably compose into complete solutions. A persistent gap separates task-native partial correctness from complete-task success. Qwen3.5-27B with thinking enabled reaches 88.7 native on `status_query` but only 35.8% complete success, while DeepSeek-v4.1-Flash reaches 89.7 native but only 34.2% complete success on the same task. Gemini 3.8 Flash at high effort similarly reaches 93.6 native on `defeat_diagnosis` while completing only 57.5% of instances. Models can therefore recover large portions of the required argumentative state while still failing to assemble a globally correct solution.

Structural richness exposes failures of global composition before local competence disappears. Complete-task success generally falls sharply in later curriculum bands. From L1–L5 to L11–L15, GPT-OSS-120B at high effort falls from 45.2% to 16.0%, Qwen3.5-27B with thinking enabled from 58.8% to 26.2%, and DeepSeek-v4.1-Flash from 57.5% to 35.0%; even Gemini 3.8 Flash at high effort declines from 88.1% to 76.5%. The gap between local and complete correctness can widen sharply: on `status_query`, Qwen3.5-27B with thinking enabled falls from 72.5% complete success with a 95.1 native score in L1–L5 to only 10.0% complete success while retaining an 84.5 native score in L11–L15. Thus, richer argumentative structure can disrupt complete state construction even when high within-task partial correctness remains.

Preference regimes change model behavior, but not according to a universal difficulty ordering. Many configurations perform markedly better under last-link than weakest-link. Gemini 3.8 Flash at high effort scores 92.9% versus 71.8%, and Qwen3.5-27B with thinking enabled 50.1% versus 28.5%. Yet task-level reversals occur: DeepSeek-v4.1-Flash, for example, performs better under weakest-link on `claim_chain` (66.7% versus 53.3%) and `defeat_diagnosis` (73.3% versus 28.3%). Weakest-link should therefore not be interpreted as uniformly harder. Because these ordering cells are generated independently, the benchmark-wide comparison is descriptive rather than a causal effect of changing the ordering on a fixed theory. Full ordering analyses are reported in Appendix D.4.

Access to a more economical intervention route does not imply that a model can exploit it. The matched `counter_argument`/`counter_argument_strict` pair holds the underlying theory and semantic objective fixed while expanding the permitted intervention language with strict-rule additions, which provide a cheaper verified construction route. Model behavior changes in opposite directions. Gemini 3.8 Flash at high effort falls from 86.7% success in `counter_argument` to 60.0% in the strict-rule-enabled condition. Qwen3.8-27B provides an even sharper contrast: from

low to xhigh effort, its overall success rises from 28.4% to 39.2% and its `counter_argument` success from 25.8% to 42.5%, while success in the strict-rule-enabled condition falls from 50.8% to 11.7%. Conversely, across the matched conditions, Qwen3.5-9B with thinking enabled rises from 20.8% to 44.2%, and Qwen3.5-35B-A3B from 15.8% to 42.5%. Constructive success therefore depends not only on whether an effective intervention exists, but also on whether a model recognizes and exploits the available intervention affordances.

6 DISCUSSION

The results point to a common limitation in current models: *local argumentative competence is more reliable than global argumentative composition*. This distinction is especially important in defeasible reasoning. Correctly identifying individual supports, attacks, statuses, or defeaters is not sufficient when the required answer depends on their joint consequences. Attacks can be defeated, preferences can determine whether an attack succeeds, and changes to one part of a theory can propagate through multiple support and defeat relations. An answer can therefore contain many correct local judgments while still representing the overall argumentative state incorrectly. The gap between native and complete-task performance shows that this is not merely a matter of models knowing too little about individual components; integration of those components is itself a major source of failure.

The curriculum and ordering results further indicate that this integration problem depends on the structure of the argumentation problem, not simply on a single underlying notion of task difficulty. As generated theories introduce longer dependencies, interacting attackers and defenders, preference-sensitive conflicts, and multi-goal structure, complete solutions become less reliable even when local correctness remains comparatively high. Preference regimes add another form of structural demand. Under last-link, comparison can depend primarily on the final defeasible steps of an argument, whereas weakest-link can make preference information distributed across premises and defeasible rules relevant to the comparison. The resulting pattern is consistent with a greater burden on maintaining and combining preference-relevant state under some weakest-link instances. At the same time, the observed task-level reversals rule out a simple interpretation in which weakest-link is merely a harder version of last-link. Model behavior depends on the interaction between the reasoning operation and the formal regime in which it is performed.

The constructive tasks expose a different limitation. Defeasible reasoning is not only about determining the state induced by a theory; it can also require reasoning backwards from a desired argumentative outcome to an intervention that produces it. The matched counter-argument conditions show that providing an additional, cheaper intervention operation does not uniformly make this problem easier for a model. Expanding the available intervention language changes the search problem, and models differ in whether they recognize and use the newly available route. This separates two capabilities that ordinary answer-only evaluation would conflate: understanding which argumentative state should result and finding an effective sequence of operations that reaches that state. A model may be strong at evaluating a formal state while still being unreliable at constructive search over that formal system.

Taken together, the findings suggest at least two separable bottlenecks in current structured defeasible reasoning: maintaining and composing distributed argumentative information, and searching effectively over interventions that change the argumentative state. Neither bottleneck is well represented by a single benchmark-wide accuracy value. Models with similar aggregate scores can fail for different reasons, and even the strongest evaluated systems retain operation-specific weaknesses. This matters for both evaluation and training: improvement on one reasoning operation should not be assumed to imply improvement on another, and gains in local correctness should not be assumed to produce globally coherent solutions.

ArgGYM makes these distinctions testable within one executable formal environment. Because the tasks share the same ASPIC+ substrate, failures can be localized to support recovery, defeat diagnosis, semantic evaluation, preference reasoning, revision, or constructive intervention without changing the underlying argumentation formalism. The procedural generators can then produce fresh verified instances targeting a particular operation or structural regime while the frozen benchmark remains unchanged. This provides a controlled way for future work to test whether verifiable-reward training can improve a specific argumentative capability, whether those gains transfer to other opera-

tions, and whether improvements in local reasoning translate into more reliable global argumentative states.

7 LIMITATIONS

We acknowledge that our evaluation does not cover the full range of currently available models. Gemini models are accessed through the provider API, DeepSeek-v4.1-Flash through a hosted API, and the remaining systems in our panel are open-weight models. Other major proprietary API-served systems, including Anthropic Claude and OpenAI's proprietary GPT-series models, are not included. Such systems may exhibit different capability profiles, and evaluating them with ArgGYM would be an important direction for future work.

Secondly, almost all ArgGYM tasks use symbolic argumentation theories to isolate the reasoning problem and enable exact verification. This leaves open how well the same capabilities transfer to more natural language. The `formalization` task provides a controlled natural-language setting, but its descriptions are template-generated. Extending ArgGYM with more varied language in future work could help separate argumentative competence from effects of lexical choice, semantic plausibility, and world knowledge.

ArgGYM also focuses on a single structured-argumentation framework. Most tasks use grounded semantics. This provides a consistent executable foundation, but it does not establish whether the same findings would hold more broadly across alternative argumentation semantics. Extending ArgGYM in this direction would allow future work to study whether the observed model weaknesses persist under different notions of argumentative acceptance.

Finally, it is important to note that the curriculum and ordering analyses should be interpreted as diagnostic. Curriculum configurations can vary several structural properties at once, while last-link and weakest-link instances are generated independently rather than as matched versions of the same underlying theory. Requiring every theory to remain both structurally informative and semantically suitable under both ordering principles would impose a stronger constraint on generation. This would reduce generation efficiency and, more importantly, could bias the accepted instances toward a narrow set of specially constructed theories that happen to behave meaningfully under both orderings. We therefore generate the ordering conditions separately and interpret the resulting differences as systematic associations rather than isolated causal effects of the ordering principle.

8 CONCLUSION

We introduced ArgGYM, a procedural, engine-verified environment for studying structured defeasible reasoning in language models. Built on a common ASPIC+ substrate, ArgGYM decomposes defeasible reasoning into twelve analysis and constructive tasks spanning representation, support recovery, defeat diagnosis, semantic evaluation, revision, preference reasoning, and intervention. A frozen 1,440-instance benchmark provides a reproducible basis for model comparison, while the same task generators, formal semantics, and verification contracts can produce fresh instances without modifying the evaluation suite. This combination turns structured argumentation into both an evaluation setting and a source of mechanically verifiable training problems.

Our results show why this decomposition matters. Defeasible reasoning in the tested models is not well described by a single aggregate capability: models can recover locally correct components while failing to compose the full argumentative state, become less reliable as structures interact, respond differently to preference regimes, and vary in their ability to exploit available intervention routes. These findings identify global argumentative composition and constructive search as distinct challenges that final-answer accuracy alone would obscure. By combining a frozen benchmark with fresh, mechanically verified instances, ArgGYM provides a controlled basis for evaluating these weaknesses and for training future models toward more reliable reasoning under incomplete, conflicting, and revisable information.

AI USE STATEMENT

Generative AI tools were used during the preparation of this work to assist with drafting, revising, and polishing manuscript text; to support literature discovery by identifying potentially relevant prior work; and to assist with implementing, debugging, and correcting the procedural generation and evaluation code. The research questions, design of the procedural engine and its tasks, and interpretation of the results were carried out by the authors. All AI-assisted text, literature references, and code changes were reviewed and verified by the authors.

ETHICS STATEMENT

This work does not involve human subjects, personal information, or sensitive data. ArgGYM instances are procedurally generated from synthetic ASPIC+ argumentation theories, and task outcomes are computed or verified using a formal argumentation engine rather than human annotation or subjective judgment. The benchmark is intended for evaluating and studying structured defeasible reasoning in language models and for supporting research on verifiable-reward training.

REPRODUCIBILITY STATEMENT

We release the frozen 1,440-instance benchmark, procedural generators, task-specific scorers and verifiers, evaluation harness, and model configuration files used in this study. The released code supports both rescoring predictions on the frozen benchmark and generating new instances under the same task definitions, curriculum configurations, preference orderings, and verification contracts.

Appendix A.1 documents the formal backend, generation procedure, argumentation configuration, and executable verification process. Appendix D.1 reports the model identifiers and inference configurations used in the evaluation. Evaluation metadata records the frozen taskset identity and relevant prompt and scoring versions, allowing reported runs to be checked against the intended benchmark and scoring protocol.

The frozen benchmark is kept separate from procedurally generated training instances. This separation permits reproduction of the reported evaluation while also allowing independent generation of fresh verified instances without modifying or training on the frozen evaluation suite.

REFERENCES

Yamen Ajjour, Carlotta Quensel, Nedim Lipka, and Henning Wachsmuth. ArgBench: Benchmarking LLMs on computational argumentation tasks. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 21846–21874, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.1099. URL https://aclanthology.org/2026.findings-acl.1099/.

Emily Allaway and Kathleen McKeown. Evaluating defeasible reasoning in LLMs with DEFREASING. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 10540–10558, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.529. URL https://aclanthology.org/2025.naacl-long.529/.

Pietro Baroni, Martin Caminada, and Massimiliano Giacomin. An introduction to argumentation semantics. *The Knowledge Engineering Review*, 26(4):365–410, 2011. doi: 10.1017/S0269888911000166.

Federico Castagna, Isabel Sassoon, and Simon Parsons. Can formal argumentative reasoning enhance llms performances?, 2024. URL https://arxiv.org/abs/2405.13036.

Günther Charwat, Wolfgang Dvořák, Sarah A. Gaggl, Johannes P. Wallner, and Stefan Woltran. Methods for solving reasoning problems in abstract argumentation – a survey. *Artificial Intelligence*, 220:28–63, 2015. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.

2014.11.008. URL https://www.sciencedirect.com/science/article/pii/S0004370214001404.

Guizhen Chen, Liying Cheng, Anh Tuan Luu, and Lidong Bing. Exploring the potential of large language models in computational argumentation. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2309–2330, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.126. URL https://aclanthology.org/2024.acl-long.126/.

Jiangjie Chen, Qianyu He, Siyu Yuan, Aili Chen, Zhicheng Cai, Weinan Dai, Hongli Yu, Jiaze Chen, Xuefeng Li, Qiying Yu, Hao Zhou, and Mingxuan Wang. Enigmata: Scaling logical reasoning in large language models with synthetic verifiable puzzles. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 3613–3661. Curran Associates, Inc., 2025a. doi: 10.52202/085713-0128. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/058460c719f190500568b341bd06875f-Paper-Conference.pdf.

Simin Chen, Yiming Chen, Zexin Li, Yifan Jiang, Zhongwei Wan, Yixin He, Dezhi Ran, Tianle Gu, Haizhou Li, Tao Xie, and Baishakhi Ray. Benchmarking large language models under data contamination: A survey from static to dynamic evaluation. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 10080–10098, Suzhou, China, November 2025b. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.511. URL https://aclanthology.org/2025.emnlp-main.511/.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. Transformers as soft reasoners over language. In Christian Bessiere (ed.), *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pp. 3882–3890. International Joint Conferences on Artificial Intelligence Organization, 7 2020. doi: 10.24963/ijcai.2020/537. URL https://doi.org/10.24963/ijcai.2020/537. Main track.

Patrick Cooper and Alvaro Velasquez. Defab: A verifiable benchmark for defeasible abduction in foundation models, 2026. URL https://arxiv.org/abs/2606.18557.

Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.*, 77(2):321–357, September 1995. ISSN 0004-3702. doi: 10.1016/0004-3702(94)00041-X. URL https://doi.org/10.1016/0004-3702(94)00041-X.

X. Fang, Z. Li, C. Chen, and B. Liao. LLM-ASPIC+: A neuro-symbolic framework for defeasible reasoning. In *ECAI 2025: 28th European Conference on Artificial Intelligence, 25–30 October 2025, Bologna, Italy – Including 14th Conference on Prestigious Applications of Intelligent Systems (PAIS 2025), Proceedings*, pp. 1567–1574. SAGE Publications, 2026. doi: 10.3233/FAIA250981.

Gabriel Freedman, Adam Dejl, Deniz Gorur, Xiang Yin, Antonio Rago, and Francesca Toni. Argumentative large language models for explainable and contestable claim verification. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'25/IAAI'25/EAAI'25. AAAI Press, 2025. ISBN 978-1-57735-897-8. doi: 10.1609/aaai.v39i14.33637. URL https://doi.org/10.1609/aaai.v39i14.33637.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei

Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645 (8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL https://doi.org/10.1038/s41586-025-09422-z.

Mehran Kazemi, Quan Yuan, Deepti Bhatia, Najoung Kim, Xin Xu, Vaiva Imbrasaite, and Deepak Ramachandran. Boardgameqa: A dataset for natural language reasoning with contradictory information. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 39052–39074. Curran Associates, Inc., 2023. doi: 10.52202/075280-1697. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/7adce80e86aa841490e6307109094de5-Paper-Datasets_and_Benchmarks.pdf.

Mario A. Leiva, Gianvincenzo Alfano, and Gerardo I. Simari. Effective benchmarking of structured argumentation via the generation of synthetic delp knowledge bases. *Know.-Based Syst.*, 346(C), July 2026. ISSN 0950-7051. doi: 10.1016/j.knosys.2026.116080. URL https://doi.org/10.1016/j.knosys.2026.116080.

Zhaoqun Li, Chen Chen, Mengze Li, and Beishui Liao. Exploring formal defeasible reasoning of large language models: A chain-of-thought approach. *Know.-Based Syst.*, 319(C), June 2025. ISSN 0950-7051. doi: 10.1016/j.knosys.2025.113564. URL https://doi.org/10.1016/j.knosys.2025.113564.

Zhaoqun Li, Xiaotong Fang, Chen Chen, Mengze Li, and Beishui Liao. Enhancing conflict resolution in language models via abstract argumentation. *Neurocomput.*, 664(C), June 2026. ISSN 0925-2312. doi: 10.1016/j.neucom.2025.132093. URL https://doi.org/10.1016/j.neucom.2025.132093.

Sanjay Modgil and Henry Prakken. The aspic+ framework for structured argumentation: a tutorial. *Argument & Computation*, 5(1):31–62, 2014. doi: 10.1080/19462166.2013.869766. URL https://doi.org/10.1080/19462166.2013.869766.

Daphne Odekerken, AnneMarie Borg, and Matti Berthold. Demonstrating PyArg 2.0. In *Proceedings of the 7th Workshop on Advances in Argumentation in Artificial Intelligence (AI^3 2023)*, volume 3546 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2023. URL https://ceur-ws.org/Vol-3546/paper14.pdf.

Mihir Parmar, Nisarg Patel, Neeraj Varshney, Mutsumi Nakamura, Man Luo, Santosh Mashetty, Arindam Mitra, and Chitta Baral. LogicBench: Towards systematic evaluation of logical rea-

soning ability of large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13679–13707, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.739. URL https://aclanthology.org/2024.acl-long.739/.

John L. Pollock. Defeasible reasoning. *Cognitive Science*, 11(4):481–518, 1987. doi: https://doi.org/10.1207/s15516709cog1104_4. URL https://onlinelibrary.wiley.com/doi/abs/10.1207/s15516709cog1104_4.

Henry Prakken. An abstract framework for argumentation with structured arguments. *Argument & Computation*, 1(2):93–124, 2010. doi: 10.1080/19462160903564592.

Henry Prakken and Giovanni Sartor. Argument-based extended logic programming with defeasible priorities. *Journal of Applied Non-Classical Logics*, 7(1–2):25–75, 1997. doi: 10.1080/11663081.1997.10510900.

Chengwen Qi, Ren Ma, Bowen Li, He Du, Binyuan Hui, Jinwang Wu, Yuanjun Laili, and Conghui He. Large language models meet symbolic provers for logical reasoning evaluation. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=C25SgeXWjE.

Brian Rabern, Philipp Mondorf, and Barbara Plank. Logicskills: A structured benchmark for formal reasoning in large language models, 2026. URL https://arxiv.org/abs/2602.06533.

R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1):81–132, 1980. ISSN 0004-3702. doi: https://doi.org/10.1016/0004-3702(80)90014-4. URL https://www.sciencedirect.com/science/article/pii/0004370280900144. Special Issue on Non-Monotonic Logic.

Rachel Rudinger, Vered Shwartz, Jena D. Hwang, Chandra Bhagavatula, Maxwell Forbes, Ronan Le Bras, Noah A. Smith, and Yejin Choi. Thinking like a skeptic: Defeasible inference in natural language. In Trevor Cohn, Yulan He, and Yang Liu (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4661–4675, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.418. URL https://aclanthology.org/2020.findings-emnlp.418/.

Jayanta Sadhu, Sayem Shahad, and Kenneth Marino. Derelab: Probing defeasible reasoning and confirmation bias in llms with a generative benchmark, 2026. URL https://arxiv.org/abs/2608.30413.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Zafir Stojanovski, Oliver Stanley, Joe Sharratt, Richard Jones, Abdulhakeem Adefioye, Jean Kaddour, and Andreas Köpf. Reasoning gym: Reasoning environments for reinforcement learning with verifiable rewards. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference. Curran Associates, Inc., 2025. doi: 10.52202/085713-1913. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/528d56195a2c77c808494c86fa7c77ad-Paper-Datasets_and_Benchmarks_Track.pdf.

Ilias Tachmazidis, Sotiris Batsakis, and Grigoris Antoniou. Benchmarking defeasible reasoning with large language models – initial experiments and future directions, 2024. URL https://arxiv.org/abs/2410.12509.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. ProofWriter: Generating implications, proofs, and abductive statements over natural language. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*,

pp. 3621–3634, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.317. URL https://aclanthology.org/2021.findings-acl.317/.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Benjamin Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Sreemanti Dey, Shubh-Agrawal, Sandeep Sandha, Siddartha Naidu, Chinmay Hegde, Yann LeCun, Tom Goldstein, Willie Neiswanger, and Micah Goldblum. Livebench: A challenging, contamination-limited llm benchmark. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 91595–91631, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/e4a46394ba5378b3f9a186a5b4c650d1-Paper-Conference.pdf.

Bryan Wilie, Samuel Cahyawijaya, Etsuko Ishii, Junxian He, and Pascale Fung. Belief revision: The adaptability of large language models reasoning. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 10480–10496, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.586. URL https://aclanthology.org/2024.emnlp-main.586/.

Yifan Xiao and Shijie Li. Argus: Argumentation-based minimal-change repair for verifiable LLM self-explanations. In Emilija Perković and Daniel Malinsky (eds.), *Proceedings of the 42nd Conference on Uncertainty in Artificial Intelligence*, volume 337 of *Proceedings of Machine Learning Research*, pp. 7536–7551. PMLR, 17–21 Aug 2026. URL https://proceedings.mlr.press/v337/xiao26a.html.

Chulin Xie, Yangsibo Huang, Chiyuan Zhang, Da Yu, Xinyun Chen, Bill Yuchen Lin, Bo Li, Badih Ghazi, and Ravi Kumar. On memorization of large language models in logical reasoning. In Kentaro Inui, Sakriani Sakti, Haofen Wang, Derek F. Wong, Pushpak Bhattacharyya, Biplab Banerjee, Asif Ekbal, Tanmoy Chakraborty, and Dhirendra Pratap Singh (eds.), *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pp. 2742–2785, Mumbai, India, December 2025a. The Asian Federation of Natural Language Processing and The Association for Computational Linguistics. ISBN 979-8-89176-298-5. doi: 10.18653/v1/2025.ijcnlp-long.148. URL https://aclanthology.org/2025.ijcnlp-long.148/.

Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning, 2025b. URL https://arxiv.org/abs/2502.14768.

Yeliang Xiu, Zhanhao Xiao, and Yongmei Liu. LogicNMR: Probing the non-monotonic reasoning ability of pre-trained language models. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 3616–3626, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.265. URL https://aclanthology.org/2022.findings-emnlp.265/.

A Benchmark Details

A.1 Formal Engine and Argumentation Configuration

PyArg backend. ArgGYM uses PyArg as its pinned formal backend for ASPIC+ computation. PyArg is an open-source Python package for computational argumentation with support for abstract argumentation, ASPIC+ argumentation theories, and assumption-based argumentation, together with algorithms for extension-based semantics (Odekerken et al., 2023). The package is distributed as `python-argumentation`; ArgGYM pins version 2.0.2 for benchmark generation and verification.

ArgGYM does not use PyArg's generic random generators to define the benchmark distribution. Instead, each task has its own procedural generator and task-specific acceptance conditions. Given

a task, curriculum configuration, argument preference ordering, set ordering, and random seed, the generator constructs candidate instances from a deterministic pseudorandom process. Structural and semantic gates reject candidates that do not exhibit the phenomenon required by the task and configuration. For an accepted candidate, the ArgGYM adapter instantiates the axioms and ordinary premises, strict and defeasible rules, conflict relations, and preferences, after which the pinned PyArg backend computes the corresponding arguments, attacks, defeats, and semantic outcomes. The benchmark represents ordinary literal conflict through signed negation and undercutting through attacks on named defeasible rules; asymmetric contrary relations are not procedurally generated. The pinned PyArg backend therefore serves as the formal-semantic oracle for ArgGYM: it computes the argumentative states and consequences against which task-specific outputs are computed or validated, while ArgGYM's task-specific generators and scorers define the structured prediction and intervention contracts used by the benchmark.

Argument preferences. The benchmark crosses two argument preference orderings with two set orderings. Under the ASPIC+ *last-link* principle, arguments are compared using their last defeasible rules, with ordinary-premise comparison applying when both arguments are strict; under *weakest-link*, relevant ordinary premises and defeasible rules throughout the arguments may contribute to the comparison (Modgil & Prakken, 2014). Preference information from an earlier defeasible step can therefore be irrelevant under last-link but decisive under weakest-link, allowing the two principles to induce different defeat relations and acceptance outcomes. Each argument preference ordering is combined with either the *elitist* or *democratic* set ordering, yielding

$$\{\text{last-link}, \text{weakest-link}\} \times \{\text{elitist}, \text{democratic}\}.$$

The selected configuration is stated explicitly whenever argument preferences can affect the solution.

Semantics. All ArgGYM tasks except `semantics_query` use grounded semantics for acceptance. Grounded semantics has a unique extension, so each theory induces a single canonical acceptance state rather than requiring a choice among multiple extensions or acceptance modes (Dung, 1995; Baroni et al., 2011). At the level of abstract argumentation, acceptance under grounded semantics is also polynomial-time decidable (Charwat et al., 2015). These properties make grounded semantics a natural default for exact and repeated verification: each theory has a deterministic acceptance target that can be computed efficiently.

The `semantics_query` task deliberately broadens this setting. It probes grounded, eager, and stable semantics, as well as credulous and skeptical acceptance under preferred semantics. Stable cases include theories with no stable extension, requiring models to distinguish the absence of a stable extension from the non-acceptance of a queried claim.

Executable verification. ArgGYM separates the evaluated model from the formal engine. During instance generation and freezing, the engine computes or validates the formal state required by each task, and the state needed by the corresponding scorer is stored in the frozen row. Predictions can therefore be rescored independently without regenerating the instance or relying on a language-model-produced reference answer.

Verification follows the contract of each task. The non-constructive tasks are evaluated against structured targets that are either computed directly by the formal engine or constructed during generation and subsequently validated against engine-computed states and task-specific structural conditions. For `formalization`, the submitted ASPIC+ directives are additionally executed. The explicitly queried claim statuses contribute to the task-native behavioral score, while complete-task success requires behavioral equivalence with the intended theory over every literal named by the reference theory. Exact reconstruction of the reference directives is not required. For the six constructive tasks, the submitted directives are parsed, checked for task-specific legality, applied to the base theory, and evaluated by the formal engine. A construction succeeds when all requested target statuses are obtained and the resulting theory remains consistent. Correctness therefore depends on the formal object produced or induced by the answer.

Frozen-gold re-execution audit. As an additional safeguard against serialization, scoring, or ordering-propagation errors, we separately re-executed the complete 1,440-instance frozen benchmark from its stored formal state using ArgGYM and PyArg 2.0.2 while explicitly supplying each

instance's recorded argument and set ordering. The audit covered all twelve tasks and all four last-link/weakest-link $\times$ democratic/elitist configurations. Every frozen reference answer or verified witness reproduced complete-task success under its corresponding task contract. We observed zero task-decoding failures, zero reference rescoring failures, zero ordering-propagation failures at either the task-payload or formal-framework level, and zero engine-invariant failures. This re-execution audit establishes internal consistency of the frozen benchmark, its serialized formal states, task-specific verification paths, and ordering propagation; it is not an independent second implementation of the underlying ASPIC+ semantics.

Consistency and strict-rule convention. For constructive tasks, consistency is evaluated on the recomputed argumentative state: a result is treated as inconsistent if a literal and its signed contrary are both justified. The presence of conflicting ordinary premises in the theory does not by itself constitute failure unless it induces such jointly justified conclusions. Strict rules, including model-added strict rules, are evaluated as explicitly declared under the pinned PyArg semantics; ArgGYM does not automatically close the submitted rule set under transposition or contraposition.

Minimality and intervention efficiency. For the six constructive tasks, ArgGYM records a task-specific intervention minimum to encourage concise solutions. During generation, each task constructs a candidate space of admissible interventions and establishes the smallest number of directives that achieves the required semantic outcome within that space. Depending on the task and instance, this is established through exact subset or combination search or through task-specific decomposition and lower-bound or necessity checks. Instances for which the generator reports that minimality was not established are excluded during freezing.

The resulting `min_directives` value is therefore a verified minimum within the generator-defined candidate space for that instance. It is not a claim of global minimality over every syntactically legal intervention. A model may submit a different legal intervention, including one not used to establish the recorded minimum, because its answer is evaluated by execution.

For a successful intervention, let m denote the recorded candidate-space minimum and n the number of submitted directives. Its native score is

$$0.5 + 0.5 \min\left(1, \frac{m}{n}\right).$$

A successful intervention with $n \leq m$ receives full native credit, while a successful intervention with $n > m$ receives lower credit. If $n > 2m$, the submission is classified as a failed construction before engine execution. Thus semantic success determines correctness within the permitted intervention size, while the native score distinguishes more economical successful constructions.

A.2 Procedural Generation and Curriculum

Within the pinned benchmark configuration, generation is deterministic given

$$(\tau, \ell, p, c, s),$$

where τ is the task, ℓ the curriculum configuration, p the argument preference ordering, c the set ordering, and s the random seed. In the implementation, the pair (p, c) is represented by a single combined ordering configuration $o = (p, c)$, corresponding to one of the four last-link/weakest-link $\times$ elitist/democratic combinations. These coordinates initialize a task-specific deterministic pseudo-random process. A generator may construct and reject several internal candidates before producing the instance associated with a seed. Candidates are retained only when task-specific structural predicates and engine-computed outcomes confirm that the intended reasoning phenomenon is present.

When the frozen benchmark is constructed, ArgGYM considers each task–level–ordering cell separately and scans seeds until the required number of accepted instances is obtained, subject to a fixed search bound. The accepted seeds and generator rejection statistics are recorded. Thus randomness provides diversity within a controlled task distribution, while deterministic seeding, explicit rejection criteria, and version pinning make the resulting benchmark reproducible.

The fifteen curriculum configurations manipulate task-relevant structural features rather than a single global complexity variable. Depending on the task, these include derivation depth, branching,

numbers and interactions of attackers and defenders, mixtures of strict and defeasible inference, distractor material, preference-sensitive conflicts, junctions between support chains, and interactions among multiple target claims. The curriculum is intended to expose progressively richer reasoning structures rather than merely increase input length. Accordingly, level number should be interpreted as curriculum progression and not as a guarantee of monotonic empirical difficulty.

A.3 Task Definitions

ArgGYM decomposes structured defeasible reasoning into twelve benchmark tasks. The tasks differ in both the formal operation required from the model and the form in which correctness can be verified. Six tasks produce structured analyses or formalizations, while six require executable interventions whose consequences are recomputed by the formal engine. The descriptions below specify the reasoning problem associated with each task; detailed scoring and verification procedures are given in Appendix B.

`formalization`. The model receives a controlled natural-language description of an argumentation theory together with the statuses that a correct formalization must induce on a set of queried claims. The description is procedurally realized from pools of sentence templates that express the underlying argumentation-theoretic distinctions, including ordinary premises, axioms, strict and defeasible inference, conflicts, and preferences, using both technical and paraphrastic formulations. The model must translate this description into executable ASPIC+ directives containing the appropriate premises or axioms, rules, and preference information. The submitted theory is then executed. Its agreement on the explicitly queried claims contributes to the task-native behavioral score, while complete-task success requires the submitted theory to reproduce the reference theory's engine-derived statuses over every literal named by the reference theory. Thus a behaviorally equivalent formalization may succeed without exactly reconstructing the generator's reference directives. The task-native score additionally measures structural and type-sensitive agreement with that reference theory.

`claim_chain`. The model is given a theory and a justified target claim and must recover the complete support derivation responsible for deriving that claim. The required support structure may branch at higher curriculum configurations when a rule depends on multiple supporting lines. The answer therefore consists of the premises and inference rules in the target's generated support derivation, ordered from supporting premises toward the target. Because independent branches need not have a unique textual serialization, the scorer does not require the particular order in which the generator constructed the derivation. Instead, it accepts any ordering consistent with the derivational dependencies: each rule must appear only after the premises or intermediate conclusions on which it depends. Complete-task success requires recovery of all and only the required support directives together with a dependency-consistent ordering.

`defeat_diagnosis`. The model is given a claim that is not justified and must determine its status and identify every relevant point at which its support fails. A diagnosis records the attacked component, the defeater, and whether the failure is due to undermining, rebutting, or undercutting; selected instances additionally require identifying cases in which an apparent attack on the defeater is itself defeated. The task therefore requires explaining the dialectical structure responsible for non-acceptance.

`status_query`. Given an ASPIC+ theory and a set of queried literals, the model assigns each literal one of JUSTIFIED, OVERRULED, or UNDECIDED under grounded semantics. Complete-task success requires the status of every queried literal to match the engine-derived state. Generator constraints ensure that all three statuses occur among the queried claims and limit label imbalance, reducing the usefulness of majority-label shortcuts.

`semantics_query`. This task extends claim-status reasoning beyond the benchmark's default grounded setting. The model must determine claim acceptance under the semantics and reasoning mode stated in the instance. The task covers grounded and eager semantics, stable semantics, and credulous or skeptical acceptance under preferred semantics; stable instances may also contain no

stable extension. The model must therefore distinguish changes in semantic interpretation from changes in the underlying argumentation theory.

perturbation. The model is given an argumentation theory together with a specified update and must identify which claims of the original theory change status after the update, as well as the new status of each changed claim. The task therefore requires both detecting the affected claims and computing the downstream consequences of the intervention across the argumentative structure. During generation, the engine computes claim statuses before and after the update. The gold answer consists of the claims whose statuses differ between these two states together with their post-update statuses. Generated instances also retain claims whose statuses survive the perturbation, providing unchanged structure against which spurious change predictions can be distinguished.

preference_construction. The model must add preference statements that transform the current theory into a requested configuration of claim statuses. No new premises or inference rules may be introduced: the intervention is restricted to legal preferences over ordinary premises and defeasible rules. Submitted preferences are executed together and the resulting argumentative state is recomputed, allowing the task to test inverse reasoning about how preference relations alter defeat and acceptance. Successful solutions must satisfy all requested statuses while preserving consistency.

counter_argument. The model starts from a theory in which a target claim is justified and must construct an intervention that makes the opposing claim justified and the original claim overruled. The permitted intervention language includes new defeasible rules, preferences over ordinary premises or defeasible rules, and the introduction of the contrary of an existing ordinary premise for undermining. Arbitrary new positive premises and new axioms are not permitted, and this task variant does not permit the addition of strict rules. The submitted directives are executed against the original theory, so correctness is determined by the argumentative state they induce.

counter_argument_strict. This task forms a matched intervention-language ablation with the `counter_argument` task. For each matched configuration, both present the same underlying argumentation theory and require the same semantic outcome: the opposing claim must become justified and the original claim overruled. The only manipulated factor is the permitted intervention language. `counter_argument_strict` permits strict-rule additions, whereas `counter_argument` removes that operation while retaining the remaining legal intervention types. Their comparison therefore isolates model behavior under the availability of strict-rule construction rather than comparing two independently generated reasoning problems.

The paired construction is designed so that permitting strict inference provides a cheaper verified route without changing the theory being solved. Across all 120 frozen matched pairs, the recorded candidate-space minimum is lower in the strict-rule-enabled condition: it is lower by one directive in 112 pairs and by two directives in the remaining eight. In the ordinary condition, the recorded minima are 3 directives for 40 instances, 4 for 48, 5 for 24, and 6 for 8; in the strict condition, they are 2 directives for 48 instances, 3 for 40, 4 for 24, and 5 for 8. The contrast therefore tests model behavior when a cheaper verified intervention route becomes available. As for the other constructive tasks, success is determined by the recomputed argumentative state and consistency, not by reproducing the stored reference intervention.

attack. The model receives a theory in which a designated target is justified and must modify the theory so that the target becomes overruled. A successful answer must therefore construct an effective attack and ensure that it survives the resulting preference and defeat relations strongly enough to change the target's semantic status. The submitted intervention is checked for task-specific legality and evaluated by executing it against the theory.

defence. The model receives an attacked target and must construct an intervention that makes the target justified. This may require countering or disabling effective attackers, strengthening relevant support, or otherwise altering the argumentative configuration within the operations permitted by the task. Success is determined from the recomputed semantic state rather than from matching a particular defensive strategy.

`attack_defense`. The model must satisfy two semantic objectives within a single intervention: one designated target must become overruled while another becomes justified. The task therefore requires coordinating offensive and defensive modifications whose consequences can interact within the same argumentation theory. All requested postconditions must hold simultaneously and the resulting theory must remain consistent.

B TASK VERIFICATION

Each task returns two quantities: a task-native score in $[0, 1]$ and a Boolean complete-task success indicator. Complete-task success is the primary quantity used for comparisons across tasks. Native scores retain task-specific partial-credit information and are therefore interpreted only within each task.

Pair-valued prediction tasks. The `status_query`, `semantics_query`, and `perturbation` tasks are scored as key–status maps. For a predicted map P and gold map G, a true positive is a key whose submitted status matches the corresponding gold status exactly. Precision and recall are

$$\mathrm{Prec} = \frac{\mathrm{TP}}{|P|}, \qquad \mathrm{Rec} = \frac{\mathrm{TP}}{|G|},$$

with the usual harmonic-mean F_1. A key assigned multiple contradictory statuses does not count as correct. The native score is this key–status pair F_1, and complete-task success requires exact equality of the predicted and gold maps. For `status_query`, keys are queried claims; for `semantics_query`, keys are claim–semantics pairs; and for `perturbation`, keys are claims whose status changes after the specified update.

`claim_chain`. Let G be the set of directives in the target support derivation and let the submitted answer contain n directive lines. Content precision and recall are computed from the overlap between submitted and required directives, with repeated submitted directives still counting against precision. Their harmonic mean gives F_1^{content}. The scorer additionally computes

$$q = \frac{\text{number of submitted rules appearing after all of their antecedents}}{\text{number of submitted rules}},$$

taking $q = 1$ when no rule is submitted. The native score is

$$F_1^{\mathrm{content}} q.$$

Complete-task success requires all and only the required support directives and a dependency-consistent ordering; it does not require the particular ordering used internally by the generator.

`defeat_diagnosis`. The scorer compares predicted failure points with the generated diagnosis using the pair

$$(\texttt{defeated_at}, \texttt{defeater})$$

as the record key and requires the attack type (`undermine`, `undercut`, or `rebut`) to match for that record to count as correct. Let F_1^{diag} denote the resulting failure-point F_1, and let I_{status} indicate whether the submitted status of the target claim is correct. When no `survives_because` explanation is required, the native score is

$$0.85F_1^{\mathrm{diag}} + 0.15I_{\mathrm{status}}.$$

For instances requiring higher-order survival explanations, let F_1^{surv} denote F_1 over the required `survives_because` records. The native score becomes

$$0.60F_1^{\mathrm{diag}} + 0.15I_{\mathrm{status}} + 0.25F_1^{\mathrm{surv}}.$$

Complete-task success requires the target status, all failure points and their attack types, and all required survival explanations to match exactly. If an instance requires no `survives_because` explanation but the submitted answer supplies one, the submission receives native score 0 and cannot achieve complete-task success.

`formalization`. A submitted formalization is parsed as an executable ASPIC+ theory and run under the ordering specified by the instance. Let B be the pair-level F_1 between the statuses induced by the submitted theory and the specified statuses of the explicitly queried claims. Structural agreement is measured separately by an alpha-renaming-invariant multiset comparison of the submitted and reference directives; let its F_1 be S.

When the reference theory contains type-sensitive axiom or strict-rule decisions, let T denote the corresponding type-decision F_1. Its recall is computed against the axiom and strict-rule decisions in the reference theory, while its precision also penalizes additional axiom or strict-rule directives introduced by the submitted theory. The native score is

$$0.4B + 0.6S$$

when no such type-sensitive decisions are present, and

$$0.25B + 0.35S + 0.40T$$

otherwise.

Complete-task success is determined by a broader behavioral-equivalence check. The scorer collects every atom named by the reference theory through its premises, axioms, and rule antecedents or consequents, evaluates both polarities of those atoms, and requires the submitted theory to induce the same engine-computed status as the reference theory for every such literal. The queried claims must therefore also be correct, but matching only the queried statuses is insufficient. Exact reconstruction of the reference directive multiset, rule names, or rule grouping is not required.

Constructive intervention tasks. The six intervention tasks—`preference_construction`, `counter_argument`, `counter_argument_strict`, `attack`, `defence`, and `attack_defense`—share an execution-based scorer. Submitted directives are parsed and checked against task-specific legality constraints. The retained intervention is applied to the base theory, and the argumentative state is recomputed under the instance's ordering and semantics.

Complete-task success requires every requested postcondition to hold simultaneously and the resulting theory to remain consistent. Let m be the recorded candidate-space minimum and n the number of submitted readable directives. Successful interventions with a known minimum receive

$$0.5 + 0.5 \min\left(1, \frac{m}{n}\right).$$

Thus a successful intervention using at most m directives receives native score 1, while longer successful constructions receive decreasing efficiency credit. A submission using more than $2m$ directives is classified as failed before execution. When a construction does not achieve complete success, partial credit is computed from semantic progress relative to the argumentative state before the intervention. Consistent unsuccessful constructions receive at most 0.25. For each requested goal that was not already satisfied in the base theory, the progress value is 1 if the intervention achieves the requested status and 0.4 if it moves the goal to UNDECIDED from a different initial status when either JUSTIFIED or OVERRULED was requested. Thus, leaving a goal unchanged at UNDECIDED does not receive deadlock credit, and a goal that was already satisfied before the intervention does not receive free positive credit. A goal that is already satisfied in the base theory is excluded from the progress average if it remains satisfied, but contributes zero if the intervention breaks it.

For an OVERRULED goal with derived support subgoals, an otherwise unsatisfied goal may instead receive $0.3u$, where u is the fraction of relevant subgoals whose support has been defeated by the intervention. A subgoal is included in this calculation only if it was initially JUSTIFIED or its status was changed by the submitted intervention, preventing unchanged irrelevant subgoals from contributing credit. Progress is averaged across the applicable requested goals, and the native score of a consistent unsuccessful construction is 0.25 times this average. Inconsistent resulting theories receive no partial credit.

Every readable submitted directive counts toward n, including duplicates and directives subsequently rejected by legality checks. Unparseable responses are assigned zero. The recorded minimum is a verified minimum within the generator-defined candidate space for the instance, not a claim of global minimality over every syntactically legal intervention.

Scoring interpretation. Because the native scores encode different notions of partial correctness across tasks, they are not averaged into a benchmark-wide native score. Benchmark-wide performance is summarized using complete-task success; native scores are used only for within-task diagnostics and comparisons.

C Representative Benchmark Instances

The examples below are drawn from frozen benchmark instances and are presented in abridged form for readability. Omitted material consists of generated theory lines or detailed parser-level legality instructions; the evaluated models received the complete original prompts. Constructive-task reference answers are verified witnesses rather than strings that the model must reproduce.

C.1 Grounded Claim-Status

The first example illustrates the basic acceptance-query interface. Although relatively small, the instance contains strict and defeasible inference, conflicting arguments, and premise preferences.

Task	`status_query`
Level	2
Ordering	last-link democratic
Semantics	grounded
Output type	claim-status map

```
The following is a defeasible argumentation theory, evaluated under
grounded semantics with the last-link democratic strength ordering.

[premise: ko9]
[axiom: mp0]
[premise: -lp3]
[premise: kk6]
[premise: lp3]

... additional generated premises and rules omitted here ...

[defeasible vo8: ko9 => ls2]
[defeasible mi1: ls2 => ik2]
[defeasible hy5: lq8 => dv8]
[defeasible hi2: gn5 => -ls2]
[defeasible ve8: mk7 => lq8]

[prefer_premise: -mk7 > mk7]
[prefer_premise: -lp3 > lp3]

State the status of each of the following claims:
ls2, mo0, ik2, dv8, mk7, ho7.

Possible statuses: justified, overruled, undecided.
A claim is justified when some argument for it is accepted,
overruled when every argument for it is defeated, and undecided
otherwise.

Answer format: one line per claim, written as 'claim: status'.
```

The engine-derived answer is

```
ls2: undecided
mo0: justified
ik2: undecided
dv8: overruled
mk7: overruled
ho7: justified
```

C.2 DEFEAT DIAGNOSIS

In this question, the model must identify *where* each possible support route for a claim fails, *how* it is attacked, and, where required, why the defeater remains effective.

Task	`defeat_diagnosis`
Level	5
Ordering	last-link democratic
Semantics	grounded
Output type	structured diagnostic record
Target claim	`im1`

A presentation excerpt of the generated theory is:

```
[premise: al9]
[premise: fl4]
[premise: -fl4]
[premise: br3]
[premise: fv8]

... additional generated premises and rules omitted here ...

[defeasible ta7: iq0 => im1]
[defeasible so1: bw7 => lp5]
[defeasible fu0: -ik2 => bw7]
[defeasible ve3: mo9 => bv5]
[defeasible ga3: bv5 => cs7]

[prefer_premise: -fl4 > fl4]
[prefer_premise: -ik2 > ik2]
[prefer_rule: vo7 > ga3]

The claim im1 is not justified.
State its status, and identify every point at which its support fails.

For defeated_at:
  - use the attacked ordinary premise for an undermine,
  - the attacked defeasible rule name for an undercut,
  - the attacked conclusion literal for a rebut.

For defeater:
  - use the attacking premise literal for an undermine,
  - otherwise use the defeasible rule whose conclusion performs the attack.

Append
  survives_because: <rule>
only when the listed defeater is itself attacked but that attack is defeated.

Answer format:
   first line: status: overruled or status: undecided
   then one line per failure point:
   defeated_at: <target>; defeater: <defeater>;
   kind: undermine|undercut|rebut
```

The generator-constructed, engine-validated reference diagnosis is

```
status: overruled
defeated_at: fl4; defeater: -fl4; kind: undermine
defeated_at: so1; defeater: ze6; kind: undercut;
             survives_because: si5
defeated_at: cs7; defeater: vo7; kind: rebut;
             survives_because: pu0
```

This example exposes information that a single claim-status label would hide. The model must distinguish undermining, undercutting, and rebutting, locate the attacked component on each failed support route, and represent a higher-order case in which an apparent counterattack does not neutralize the operative defeater.

C.3 Strict Counterargument Ablation

This instance illustrates strict-rule-enabled condition in the matched counter-argument ablation. Its paired `counter_argument` instance uses the same base theory and semantic objective; the strict variant differs by additionally permitting strict-rule additions. The model must therefore construct a small intervention that changes the argumentative state while having access to an intervention form unavailable in the paired non-strict arm.

Task	`counter_argument_strict`
Level	6
Ordering	weakest-link elitist
Semantics	grounded
Output type	executable directive list
Recorded minimum	2 directives

```
The following is a defeasible argumentation theory, evaluated under
grounded semantics with the weakest-link elitist strength ordering.

[premise: gv9]
[premise: hq8]
[premise: jr5]
[premise: -kp2]
[premise: -ay2]
[axiom: bv4]

... additional generated premises and rules omitted here ...

[defeasible fy9: by3 AND hu0 => lq4]
[defeasible za8: -kp2 => gu6]
[strict hu2: lq4 -> au0]
[defeasible ji8: bt8 => by3]
[defeasible su9: km2 AND gq4 => by7]
[strict xi0: bv4 -> kt8]
[defeasible ha8: cm8 => -ju8]

[prefer_premise: -kp2 > kp2]
[prefer_rule: mi6 > ha8]

The claim au0 is currently justified.
What is the minimal set of directives that makes -au0 justified
and au0 overruled?

Permitted additions, written exactly in these forms:
   [defeasible <name>: <antecedent> => <consequent>]
   [strict <name>: <antecedent> -> <consequent>]
   [prefer_rule: <defeasible rule name> > <defeasible rule name>]
   [prefer_premise: <ordinary premise literal> >
                    <ordinary premise literal>]
   [premise: -<literal>]

New axioms may not be added.

... additional parser-level syntax and legality constraints omitted here ...

The answer must be minimal: one using more than twice the fewest
directives that work scores zero. A directive that cannot be read
at all scores the whole answer zero. An answer that reaches every
```

Table 3: **Model and inference configurations.** T denotes temperature, p top-p, k top-k, and PP presence penalty. Max. out. is the maximum generated-token budget.

Model	Reasoning	Sampling	Seed	Max. out.	Context
Gemma					
`google/gemma-4-E2B-it`	off / on	$T = 1.0,\ p = .95,\ k = 64$	0	65536	131072
`google/gemma-4-E4B-it`	off / on	$T = 1.0,\ p = .95,\ k = 64$	0	65536	131072
`google/gemma-4-26B-A4B-it`	off / on	$T = 1.0,\ p = .95,\ k = 64$	0	65536	262144
`google/gemma-4-31B-it`	off / on	$T = 1.0,\ p = .95,\ k = 64$	0	65536	262144
GPT-OSS					
`openai/gpt-oss-20b`	low / medium / high	$T = 1.0,\ p = 1.0$	0	65536	131072
`openai/gpt-oss-120b`	low / medium / high	$T = 1.0,\ p = 1.0$	0	65536	131072
Qwen					
`Qwen/Qwen3.5-0.8B`	off	$T = 1.0,\ p = 1.0,\ k = 20,\ \mathrm{PP} = 2.0$	0	65536	262144
	on	$T = 1.0,\ p = .95,\ k = 20,\ \mathrm{PP} = 1.5$	0	65536	262144
`Qwen/Qwen3.5-2B`	off	$T = 1.0,\ p = 1.0,\ k = 20,\ \mathrm{PP} = 2.0$	0	65536	262144
	on	$T = 1.0,\ p = .95,\ k = 20,\ \mathrm{PP} = 1.5$	0	65536	262144
`Qwen/Qwen3.5-4B`	off / on	$T = 1.0,\ p = .95,\ k = 20,\ \mathrm{PP} = 1.5$	0	65536	262144
`Qwen/Qwen3.5-9B`	off / on	$T = 1.0,\ p = .95,\ k = 20,\ \mathrm{PP} = 1.5$	0	65536	262144
`Qwen/Qwen3.5-35B-A3B`	off / on	$T = 1.0,\ p = .95,\ k = 20,\ \mathrm{PP} = 1.5$	0	65536	262144
`Qwen/Qwen3.5-27B`	off / on	$T = 1.0,\ p = .95,\ k = 20,\ \mathrm{PP} = 1.5$	0	65536	262144
`Qwen/Qwen3.8-27B`	low / medium / xhigh	$T = 1.0,\ p = .95,\ k = 20,\ \mathrm{PP} = 0$	0	65536	131072
DeepSeek					
`deepseek-v4.1-flash`	high	$T = 1.0,\ p = .95$	–	65536	–
Gemini					
`gemini-2.5-pro`	low / medium / high	provider defaults	–	65536	–
`gemini-3.5-flash-lite`	low / medium / high	provider defaults	–	65536	–
`gemini-3.1-pro-preview`	low / medium / high	provider defaults	–	65536	–
`gemini-3.8-flash`	low / medium / high	provider defaults	–	65536	–

```
goal while leaving the theory inconsistent also scores zero.

Answer format: one directive per line.
```

One verified reference intervention is

```
[strict cs: gv9 -> -au0]
[strict u1: gv9 -> -ji8]
```

The scorer does not require these particular rule names or this particular construction. It parses the submitted directives, applies them to the base theory, recomputes the argumentative state, and checks whether `-au0` is justified and `au0` is overruled while the resulting theory remains consistent. The instance records a two-directive minimum within the generator's verified candidate space.

D Detailed Evaluation Results

D.1 Model and Inference Configurations

Table 3 reports the inference configurations used in the evaluation. We evaluate 18 distinct model variants under 42 model–reasoning configurations. Reasoning configurations for the same model share decoding parameters unless otherwise indicated. For locally served models, context denotes the `max_model_len` configured in the corresponding vLLM serving profile. A dash indicates a parameter that is not explicitly configured by the evaluation harness.

For Gemma and Qwen3.5, thinking is controlled through the model chat template. GPT-OSS and Gemini 2.5 Pro/3.5 Flash Lite receive explicit reasoning-effort settings. Gemini 3.1 Pro Preview and Gemini 3.8 Flash use the provider's `thinking_level` control. Qwen3.8-27B uses explicit `reasoning_effort` values of `low`, `medium`, and `xhigh`. DeepSeek-v4.1-Flash is evaluated with `reasoning_effort=high` through its chat-template configuration.

The Qwen configurations additionally set `min_p=0.0` and `repetition_penalty=1.0`; these neutral settings are omitted from the table for readability.

Table 4: **Analysis and evaluation tasks.** Each cell reports complete-task success (%) | task-native score $\times 100$.

Model	Reasoning	Formal.	Chain	Diagnosis	Status	Semantics	Perturb.
Gemma							
`google/gemma-4-E2B-it`	off	0.8|1.0	2.5|3.5	5.0|13.1	1.7|47.6	0.0|38.0	0.0|3.7
	on	0.8|1.5	5.0|5.3	0.0|10.0	4.2|52.1	0.0|42.2	0.0|4.8
`google/gemma-4-E4B-it`	off	4.2|16.8	2.5|3.3	5.8|23.5	11.7|69.4	2.5|58.6	0.8|12.7
	on	4.2|25.1	6.7|7.9	9.2|26.1	10.8|71.4	0.0|56.8	0.8|16.2
`google/gemma-4-26B-A4B-it`	off	7.5|43.2	2.5|3.3	9.2|35.5	15.0|75.1	4.2|64.5	1.7|22.8
	on	10.8|44.8	15.8|18.8	10.8|53.3	10.8|72.9	10.0|66.3	2.5|36.8
`google/gemma-4-31B-it`	off	14.2|64.9	24.2|24.6	7.5|42.2	6.7|74.9	10.0|71.6	3.3|46.1
	on	50.8|78.9	30.8|31.6	25.8|69.8	20.0|77.4	22.5|81.8	6.7|58.3
GPT-OSS							
`openai/gpt-oss-20b`	low	5.8|14.4	9.2|12.7	8.3|19.0	4.2|41.4	0.8|39.2	0.0|2.0
	medium	5.8|50.1	31.7|35.0	14.2|38.9	3.3|65.7	5.0|57.1	0.0|14.9
	high	10.8|66.1	40.0|42.0	16.7|48.1	5.0|63.6	1.7|57.6	0.0|14.5
`openai/gpt-oss-120b`	low	7.5|33.5	27.5|29.2	7.5|27.0	5.8|67.5	3.3|46.7	0.0|10.3
	medium	9.2|60.0	49.2|50.2	10.8|47.0	15.0|77.3	10.8|67.1	0.0|24.5
	high	14.2|75.2	45.0|45.9	20.0|64.0	15.8|78.1	13.3|70.6	5.0|34.5
Qwen							
`Qwen/Qwen3.5-0.8B`	off	0.0|0.2	0.0|0.0	0.0|3.9	0.0|12.0	0.0|4.8	0.0|0.9
	on	0.0|0.3	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0
`Qwen/Qwen3.5-2B`	off	1.7|1.9	0.8|1.2	0.8|5.8	0.0|24.1	0.0|23.0	0.0|1.0
	on	0.8|1.8	2.5|3.5	5.8|8.7	0.8|6.6	0.0|15.2	0.0|0.2
`Qwen/Qwen3.5-4B`	off	5.8|26.8	31.7|37.8	12.5|41.9	12.5|79.6	11.7|63.6	1.7|23.8
	on	7.5|26.2	50.0|52.5	20.0|53.1	20.8|75.8	8.3|65.5	2.5|24.8
`Qwen/Qwen3.5-9B`	off	8.3|36.0	38.3|42.1	16.7|55.2	23.3|86.5	7.5|66.6	5.8|41.9
	on	10.0|37.6	42.5|47.4	22.5|64.8	20.0|81.3	13.3|73.3	3.3|47.2
`Qwen/Qwen3.5-35B-A3B`	off	15.8|43.3	36.7|41.9	20.0|65.2	29.2|89.6	30.0|83.5	7.5|54.5
	on	12.5|43.9	44.2|48.2	31.7|76.6	39.2|91.9	24.2|83.0	15.8|59.8
`Qwen/Qwen3.5-27B`	off	18.3|58.5	45.8|48.0	20.0|72.5	25.8|86.2	21.7|80.0	7.5|63.8
	on	22.5|64.1	50.0|60.1	39.2|86.6	35.8|88.7	22.5|82.2	15.0|71.8
`Qwen/Qwen3.8-27B`	low	35.0|75.9	38.3|38.9	27.5|74.6	5.8|69.2	5.8|59.4	7.5|43.1
	medium	30.8|77.1	46.7|49.7	35.0|80.1	6.7|71.9	9.2|68.6	1.7|52.8
	xhigh	34.2|86.0	61.7|63.2	50.0|92.3	44.2|91.4	18.3|77.3	24.2|72.0
DeepSeek							
`deepseek-v4.1-flash`	high	67.5|91.5	60.0|60.1	50.8|88.2	34.2|89.7	15.8|72.2	46.7|83.0
Gemini							
`gemini-2.5-pro`	low	41.7|58.2	24.2|31.6	14.2|35.8	6.7|71.1	1.7|57.4	0.8|21.6
	medium	44.2|68.6	49.2|55.8	35.8|88.7	35.0|92.4	26.7|78.8	25.0|87.7
	high	40.0|65.4	49.2|56.9	38.3|88.4	37.5|93.2	21.7|78.3	22.5|88.8
`gemini-3.5-flash-lite`	low	21.7|46.8	32.5|33.2	16.7|34.6	16.7|66.2	10.8|65.4	0.8|25.4
	medium	58.3|85.1	38.3|39.3	22.5|73.6	23.3|83.7	14.2|73.2	3.3|41.1
	high	84.2|87.0	50.0|50.5	35.0|84.8	25.8|89.3	19.2|79.0	10.0|68.9
`gemini-3.1-pro-preview`	low	95.8|89.6	52.5|69.8	51.7|92.1	23.3|45.2	29.2|84.9	32.5|80.2
	medium	99.2|89.9	57.5|85.3	56.7|93.3	20.8|24.6	56.7|92.6	50.0|92.6
	high	97.5|90.6	58.3|92.3	60.8|94.1	32.5|61.4	76.7|96.3	64.2|95.4
`gemini-3.8-flash`	low	97.5|90.8	58.3|62.8	57.5|93.2	55.0|95.4	67.5|96.0	53.3|80.3
	medium	98.3|94.3	88.3|94.1	57.5|93.6	63.3|97.2	92.5|98.6	84.2|97.3
	high	99.2|97.5	85.8|94.3	57.5|93.6	61.7|97.1	98.3|99.8	86.7|98.3

D.2 Task-Level Performance

Here we provide the corresponding task-level results for all twelve ArgGYM tasks. Each cell reports *complete-task success (%)* followed by the *task-native score* multiplied by 100. Native scores retain task-specific semantics and should therefore be compared only within the same task.

Table 5: **Constructive intervention tasks.** Each cell reports complete-task success (%) | task-native score $\times 100$.

Model	Reasoning	Pref.	Counter	Counter+S	Attack	Defence	Attack–Def.
Gemma							
`google/gemma-4-E2B-it`	off	2.5|2.7	0.0|0.0	0.0|0.1	0.0|0.9	0.8|0.8	0.0|0.6
	on	3.3|3.5	0.0|0.2	0.0|0.2	0.0|1.2	2.5|2.5	0.0|1.0
`google/gemma-4-E4B-it`	off	12.5|14.7	0.0|2.1	0.0|1.8	0.0|3.9	2.5|4.5	0.0|2.4
	on	8.3|11.6	0.0|2.3	0.0|3.6	0.0|4.2	4.2|6.8	0.8|5.0
`google/gemma-4-26B-A4B-it`	off	30.8|33.4	2.5|4.9	4.2|6.8	0.0|4.3	20.8|22.1	10.0|15.2
	on	35.0|36.7	8.3|12.9	15.8|18.1	9.2|14.0	22.5|25.8	17.5|22.7
`google/gemma-4-31B-it`	off	40.8|42.5	5.0|8.6	15.0|16.3	8.3|10.5	28.3|31.8	9.2|15.9
	on	46.7|47.0	14.2|17.8	16.7|18.7	20.0|21.1	39.2|41.9	34.2|37.8
GPT-OSS							
`openai/gpt-oss-20b`	low	8.3|8.7	3.3|4.3	2.5|2.7	4.2|5.0	15.8|15.7	12.5|17.6
	medium	18.3|21.1	19.2|19.6	11.7|11.4	15.8|15.9	38.3|38.7	29.2|32.8
	high	23.3|25.7	28.3|26.9	24.2|22.6	18.3|18.9	40.0|41.0	46.7|49.2
`openai/gpt-oss-120b`	low	13.3|16.0	10.8|11.2	6.7|6.2	12.5|12.3	30.0|30.0	23.3|23.5
	medium	34.2|36.5	15.8|15.6	13.3|13.0	13.3|12.9	41.7|42.3	45.0|43.7
	high	56.7|57.5	30.8|29.8	19.2|18.1	13.3|14.0	60.0|59.2	54.2|54.0
Qwen							
`Qwen/Qwen3.5-0.8B`	off	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.1
	on	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0
`Qwen/Qwen3.5-2B`	off	0.8|0.8	0.0|0.0	0.0|0.0	0.0|0.2	0.0|0.0	0.0|0.3
	on	1.7|1.9	0.0|0.1	0.0|0.1	0.0|0.1	0.8|0.8	0.0|0.0
`Qwen/Qwen3.5-4B`	off	31.7|34.1	3.3|6.2	5.8|7.5	9.2|12.2	17.5|20.0	5.8|12.0
	on	43.3|44.3	21.7|21.8	21.7|21.4	17.5|19.0	30.0|31.8	25.0|27.4
`Qwen/Qwen3.5-9B`	off	40.8|41.8	10.8|14.9	9.2|13.3	5.8|10.2	25.0|27.3	22.5|25.9
	on	43.3|43.7	20.8|23.6	44.2|43.6	24.2|25.5	32.5|34.9	36.7|38.3
`Qwen/Qwen3.5-35B-A3B`	off	40.0|41.6	10.8|12.7	15.8|16.6	5.8|8.8	26.7|29.1	21.7|26.2
	on	46.7|47.1	15.8|19.9	42.5|42.8	30.8|32.4	37.5|39.6	39.2|40.9
`Qwen/Qwen3.5-27B`	off	47.5|47.7	23.3|25.4	31.7|29.2	37.5|38.2	46.7|48.3	51.7|52.8
	on	49.2|49.3	41.7|44.2	42.5|41.9	44.2|46.7	51.7|54.0	57.5|58.4
`Qwen/Qwen3.8-27B`	low	33.3|33.5	25.8|29.3	50.8|49.4	21.7|25.2	45.8|48.3	43.3|44.4
	medium	39.2|39.5	27.5|31.4	40.8|41.5	16.7|20.5	57.5|59.5	41.7|43.6
	xhigh	50.0|49.0	42.5|43.7	11.7|11.7	27.5|24.9	45.8|48.3	60.0|57.9
DeepSeek							
`deepseek-v4.1-flash`	high	60.0|59.9	34.2|35.5	13.3|13.5	32.5|32.1	54.2|54.4	70.8|67.6
Gemini							
`gemini-2.5-pro`	low	45.0|45.6	3.3|4.6	5.0|4.6	15.0|17.9	29.2|30.1	12.5|17.6
	medium	53.3|53.5	23.3|26.2	15.8|15.9	15.0|16.8	40.0|42.2	50.8|51.3
	high	52.5|52.6	25.8|28.0	4.2|5.0	20.0|21.8	41.7|43.8	45.0|46.4
`gemini-3.5-flash-lite`	low	20.0|22.9	9.2|11.2	4.2|4.9	11.7|12.9	45.0|45.4	29.2|33.9
	medium	41.7|42.7	22.5|25.0	10.8|10.7	18.3|20.7	60.8|61.6	37.5|43.4
	high	44.2|45.5	30.0|31.4	15.0|13.8	13.3|14.8	53.3|54.2	41.7|48.3
`gemini-3.1-pro-preview`	low	49.2|49.2	46.7|47.6	18.3|16.4	41.7|41.6	70.0|70.7	69.2|70.8
	medium	52.5|52.4	56.7|58.1	20.8|19.7	35.8|35.6	65.8|66.8	86.7|86.4
	high	61.7|61.7	57.5|58.8	23.3|22.5	54.2|54.3	76.7|77.7	91.7|92.0
`gemini-3.8-flash`	low	51.7|51.9	51.7|51.9	30.0|27.2	71.7|71.5	83.3|83.4	60.0|61.7
	medium	57.5|57.5	76.7|77.3	31.7|30.5	98.3|98.3	90.8|90.8	94.2|94.6
	high	63.3|63.3	86.7|87.3	60.0|58.2	99.2|99.2	92.5|92.5	97.5|97.7

D.3 Curriculum Analysis

The fifteen curriculum configurations vary task-specific structural properties such as derivation depth, branching, interacting attackers and defenders, strict and defeasible structure, distractors, and multi-goal interactions, and, for `status_query`, the number of queried literals. They are intended to provide progressively richer configurations, not a calibrated one-dimensional difficulty scale. We therefore summarize robustness in three five-configuration bands: L1–L5, L6–L10, and L11–L15. Each band contains 480 instances for a complete evaluation configuration.

Table 6: **Complete-task success across structural-richness bands.** Values are percentages.

Model	Reasoning	L1–L5	L6–L10	L11–L15
Gemma				
`google/gemma-4-E2B-it`	off	3.3	0.0	0.0
	on	4.0	0.0	0.0
`google/gemma-4-E4B-it`	off	9.6	0.8	0.2
	on	10.2	0.6	0.4
`google/gemma-4-26B-A4B-it`	off	17.5	5.6	4.0
	on	25.8	9.6	6.9
`google/gemma-4-31B-it`	off	25.8	11.0	6.2
	on	42.3	22.9	16.7
GPT-OSS				
`openai/gpt-oss-20b`	low	16.5	1.2	1.0
	medium	27.7	14.2	6.2
	high	34.2	17.1	12.5
`openai/gpt-oss-120b`	low	24.6	9.8	2.7
	medium	34.8	18.5	11.2
	high	45.2	25.6	16.0
Qwen				
`Qwen/Qwen3.5-0.8B`	off	0.0	0.0	0.0
	on	0.0	0.0	0.0
`Qwen/Qwen3.5-2B`	off	1.0	0.0	0.0
	on	2.9	0.2	0.0
`Qwen/Qwen3.5-4B`	off	26.5	7.5	3.3
	on	40.6	16.2	10.2
`Qwen/Qwen3.5-9B`	off	34.0	11.9	7.7
	on	42.5	22.1	13.8
`Qwen/Qwen3.5-35B-A3B`	off	43.3	14.2	7.5
	on	49.4	28.5	17.1
`Qwen/Qwen3.5-27B`	off	52.7	25.0	16.7
	on	58.8	32.9	26.2
`Qwen/Qwen3.8-27B`	low	40.4	26.0	18.8
	medium	37.5	27.1	23.8
	xhigh	50.0	38.8	28.7
DeepSeek				
`deepseek-v4.1-flash`	high	57.5	42.5	35.0
Gemini				
`gemini-2.5-pro`	low	27.9	14.4	7.5
	medium	48.5	31.9	23.1
	high	48.3	29.2	22.1
`gemini-3.5-flash-lite`	low	38.1	11.9	4.6
	medium	47.5	21.9	18.5
	high	54.2	30.6	20.6
`gemini-3.1-pro-preview`	low	62.7	42.9	39.4
	medium	69.8	51.9	43.1
	high	76.7	62.5	49.6
`gemini-3.8-flash`	low	76.7	60.4	47.3
	medium	83.1	76.7	73.5
	high	88.1	82.5	76.5

Table 6 reports complete-task success separately for every model–reasoning configuration. Across most configurations, success decreases in the later bands, while stronger models and higher reasoning settings generally retain more performance as structural richness increases.

D.3.1 Task-Specific Curriculum Profiles

To identify which reasoning operations are most sensitive to structural richness, Tables 7–12 report task-level performance separately for the three curriculum bands. Each task-specific band contains 40 frozen benchmark instances for a complete model–reasoning configuration. Each cell reports complete-task success (%) followed by the task-native score multiplied by 100. Native scores retain task-specific semantics and should therefore be compared only within the same task.

The task-level decomposition shows that structural richness does not affect all argumentative operations uniformly. For many analysis tasks, complete-task success falls much more sharply than the corresponding native score. For example, Qwen3.5-27B with thinking enabled reaches 72.5% complete success with a 95.1 native score on `status_query` in L1–L5, but only 10.0% complete success with an 84.5 native score in L11–L15. This pattern indicates that substantial local correct-

Table 7: **Task-specific analysis/evaluation performance for curriculum configurations L1–L5.** Each cell reports complete-task success (%) | task-native score $\times 100$.

Model	Reasoning	Formal.	Chain	Diagnosis	Status	Semantics	Perturb.
Gemma							
`google/gemma-4-E2B-it`	off	2.5|3.1	7.5|8.4	15.0|26.2	5.0|56.0	0.0|40.2	0.0|8.2
	on	2.5|4.6	15.0|15.7	0.0|16.1	12.5|65.2	0.0|47.3	0.0|9.1
`google/gemma-4-E4B-it`	off	12.5|32.0	5.0|5.0	17.5|50.7	35.0|81.5	7.5|71.0	2.5|28.0
	on	12.5|35.3	15.0|16.7	27.5|54.4	32.5|84.8	0.0|62.1	2.5|30.7
`google/gemma-4-26B-A4B-it`	off	20.0|49.0	7.5|7.5	27.5|73.8	42.5|86.9	12.5|75.2	5.0|37.5
	on	17.5|50.4	42.5|48.8	30.0|85.7	27.5|80.5	30.0|82.1	7.5|45.8
`google/gemma-4-31B-it`	off	27.5|66.0	45.0|45.0	22.5|68.0	15.0|78.8	30.0|82.8	10.0|54.5
	on	55.0|76.1	65.0|67.3	57.5|93.3	45.0|86.3	65.0|94.4	20.0|72.2
GPT-OSS							
`openai/gpt-oss-20b`	low	17.5|34.3	27.5|31.8	25.0|40.4	12.5|60.2	2.5|47.9	0.0|5.3
	medium	12.5|49.6	52.5|52.5	42.5|66.5	10.0|70.2	12.5|69.8	0.0|24.0
	high	25.0|75.1	60.0|60.0	50.0|81.6	15.0|70.4	5.0|64.1	0.0|21.2
`openai/gpt-oss-120b`	low	17.5|54.4	52.5|53.8	22.5|53.6	17.5|80.0	10.0|60.2	0.0|18.1
	medium	20.0|60.7	77.5|77.5	32.5|76.3	45.0|83.8	32.5|80.6	0.0|38.4
	high	32.5|75.1	70.0|72.4	57.5|89.5	30.0|79.9	37.5|80.1	15.0|51.5
Qwen							
`Qwen/Qwen3.5-0.8B`	off	0.0|0.5	0.0|0.0	0.0|2.6	0.0|14.9	0.0|8.7	0.0|1.2
	on	0.0|1.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0
`Qwen/Qwen3.5-2B`	off	5.0|5.6	2.5|2.9	2.5|7.5	0.0|26.8	0.0|25.9	0.0|1.7
	on	2.5|2.5	7.5|10.6	17.5|23.7	2.5|18.8	0.0|24.5	0.0|0.6
`Qwen/Qwen3.5-4B`	off	17.5|45.2	60.0|65.7	37.5|69.5	35.0|83.9	35.0|81.6	5.0|33.5
	on	17.5|49.0	87.5|87.5	57.5|86.8	50.0|87.9	22.5|75.9	7.5|38.5
`Qwen/Qwen3.5-9B`	off	25.0|59.4	80.0|84.2	50.0|87.0	50.0|89.2	22.5|77.9	17.5|55.3
	on	17.5|50.9	77.5|79.8	55.0|88.4	42.5|87.8	40.0|87.1	10.0|61.3
`Qwen/Qwen3.5-35B-A3B`	off	35.0|52.5	67.5|72.2	52.5|86.6	67.5|95.3	72.5|94.8	22.5|66.2
	on	27.5|64.1	87.5|91.6	65.0|92.3	65.0|95.0	65.0|89.0	45.0|81.6
`Qwen/Qwen3.5-27B`	off	30.0|70.2	80.0|86.6	55.0|91.4	47.5|89.8	62.5|94.4	22.5|67.2
	on	27.5|70.1	85.0|87.3	67.5|94.8	72.5|95.1	62.5|91.9	42.5|79.7
`Qwen/Qwen3.8-27B`	low	47.5|79.6	67.5|69.2	55.0|92.9	12.5|72.3	17.5|74.0	17.5|52.8
	medium	37.5|82.5	77.5|81.4	55.0|92.4	12.5|74.1	25.0|77.0	5.0|56.6
	xhigh	37.5|85.8	95.0|95.0	60.0|94.0	67.5|95.6	40.0|84.6	47.5|82.7
DeepSeek							
`deepseek-v4.1-flash`	high	85.0|91.3	90.0|90.0	60.0|92.7	52.5|90.4	37.5|81.8	50.0|82.8
Gemini							
`gemini-2.5-pro`	low	52.5|65.3	42.5|46.7	42.5|66.4	20.0|75.9	2.5|64.3	2.5|22.4
	medium	52.5|71.1	92.5|94.8	52.5|93.0	60.0|93.2	47.5|86.3	52.5|89.7
	high	52.5|72.0	77.5|79.8	67.5|95.4	75.0|97.1	50.0|85.2	47.5|87.9
`gemini-3.5-flash-lite`	low	45.0|70.1	67.5|67.5	50.0|74.1	47.5|89.7	30.0|79.2	2.5|46.2
	medium	72.5|81.0	85.0|85.0	60.0|90.1	57.5|92.2	37.5|89.1	10.0|66.6
	high	90.0|84.8	90.0|90.0	62.5|92.7	50.0|90.5	52.5|92.7	27.5|79.0
`gemini-3.1-pro-preview`	low	95.0|88.1	90.0|96.8	70.0|95.3	62.5|79.7	75.0|96.7	67.5|93.0
	medium	100.0|89.0	92.5|97.0	70.0|95.5	62.5|69.1	87.5|94.9	70.0|94.9
	high	100.0|89.0	97.5|99.4	75.0|96.3	60.0|79.8	95.0|99.2	75.0|95.8
`gemini-3.8-flash`	low	100.0|90.0	97.5|97.5	72.5|95.9	75.0|97.0	95.0|99.4	72.5|93.3
	medium	100.0|92.1	100.0|100.0	72.5|95.9	77.5|97.4	97.5|99.5	77.5|96.1
	high	100.0|98.1	100.0|100.0	72.5|95.9	77.5|97.4	100.0|100.0	82.5|96.8

ness can persist even when models increasingly fail to assemble the complete argumentative state required by the task.

Constructive tasks show a different profile. Some intervention types remain comparatively robust for stronger models, while others remain difficult or vary non-monotonically across bands. In particular, the strict-rule counter-argument condition remains challenging for many systems even when other constructive operations improve substantially. The curriculum bands should therefore not be interpreted as calibrated scalar difficulty levels: individual tasks respond differently to the structural properties varied by their generators, and occasional non-monotonic changes occur. These results characterize task-specific robustness under increasingly rich generated configurations rather than a universal difficulty progression.

Table 8: **Task-specific constructive performance for curriculum configurations L1–L5.** Each cell reports complete-task success (%) | task-native score $\times 100$.

Model	Reasoning	Pref.	Counter	Counter+S	Attack	Defence	Attack–Def.
Gemma							
`google/gemma-4-E2B-it`	off	7.5|8.2	0.0|0.1	0.0|0.2	0.0|2.2	2.5|2.5	0.0|1.4
	on	10.0|10.0	0.0|0.4	0.0|0.6	0.0|3.1	7.5|7.5	0.0|1.9
`google/gemma-4-E4B-it`	off	30.0|31.8	0.0|4.8	0.0|4.4	0.0|7.0	5.0|5.0	0.0|3.9
	on	20.0|21.4	0.0|3.6	0.0|5.8	0.0|6.9	12.5|12.5	0.0|5.2
`google/gemma-4-26B-A4B-it`	off	42.5|43.2	7.5|10.2	5.0|8.0	0.0|8.5	32.5|32.5	7.5|13.9
	on	42.5|42.5	17.5|19.5	22.5|23.1	10.0|16.1	40.0|40.0	22.5|27.5
`google/gemma-4-31B-it`	off	50.0|50.0	10.0|12.2	22.5|20.3	15.0|17.2	50.0|50.0	12.5|20.2
	on	50.0|50.0	12.5|15.2	22.5|23.4	20.0|20.6	57.5|57.5	37.5|39.6
GPT-OSS							
`openai/gpt-oss-20b`	low	25.0|25.9	10.0|12.2	5.0|5.6	12.5|13.9	35.0|34.6	25.0|27.2
	medium	32.5|34.3	32.5|32.6	10.0|10.8	35.0|30.8	55.0|54.4	37.5|41.4
	high	37.5|37.5	37.5|35.8	30.0|25.7	27.5|26.4	62.5|62.5	60.0|60.9
`openai/gpt-oss-120b`	low	37.5|38.6	22.5|21.7	17.5|15.3	27.5|23.8	40.0|39.6	30.0|28.2
	medium	45.0|45.7	25.0|24.6	15.0|13.5	30.0|26.1	55.0|55.0	40.0|39.2
	high	62.5|63.1	35.0|35.5	37.5|35.0	17.5|17.8	95.0|93.1	52.5|52.7
Qwen							
`Qwen/Qwen3.5-0.8B`	off	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.1
	on	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0
`Qwen/Qwen3.5-2B`	off	2.5|2.5	0.0|0.0	0.0|0.0	0.0|0.4	0.0|0.0	0.0|0.5
	on	5.0|5.7	0.0|0.2	0.0|0.0	0.0|0.3	0.0|0.0	0.0|0.0
`Qwen/Qwen3.5-4B`	off	40.0|40.8	10.0|13.3	10.0|11.9	17.5|21.4	37.5|36.9	12.5|18.4
	on	47.5|47.5	37.5|36.1	42.5|38.6	30.0|29.8	45.0|45.0	42.5|43.4
`Qwen/Qwen3.5-9B`	off	45.0|45.0	17.5|22.3	20.0|25.2	12.5|18.7	37.5|37.5	30.0|31.8
	on	47.5|47.5	27.5|30.1	72.5|67.5	35.0|36.1	47.5|47.5	37.5|37.9
`Qwen/Qwen3.5-35B-A3B`	off	50.0|50.0	22.5|23.9	35.0|32.3	12.5|16.9	47.5|47.5	35.0|38.3
	on	50.0|50.0	17.5|21.7	45.0|45.8	42.5|44.7	45.0|45.0	37.5|39.3
`Qwen/Qwen3.5-27B`	off	47.5|47.5	42.5|43.1	47.5|44.0	70.0|70.2	65.0|65.0	62.5|62.0
	on	50.0|50.0	47.5|49.9	55.0|54.7	55.0|56.6	72.5|72.5	67.5|67.4
`Qwen/Qwen3.8-27B`	low	30.0|28.3	30.0|31.8	50.0|48.7	32.5|34.5	70.0|70.0	55.0|53.1
	medium	35.0|34.2	12.5|15.4	42.5|42.8	22.5|25.1	80.0|80.0	45.0|46.0
	xhigh	45.0|42.0	37.5|38.4	12.5|12.5	17.5|14.9	72.5|72.5	67.5|63.1
DeepSeek							
`deepseek-v4.1-flash`	high	62.5|62.1	37.5|38.8	25.0|24.6	32.5|31.0	80.0|80.0	77.5|71.7
Gemini							
`gemini-2.5-pro`	low	50.0|49.9	10.0|10.7	10.0|8.2	32.5|36.2	55.0|54.4	15.0|20.7
	medium	52.5|52.5	17.5|19.4	15.0|14.0	17.5|20.3	65.0|65.0	57.5|57.1
	high	55.0|55.0	17.5|18.2	5.0|4.4	15.0|16.8	60.0|60.0	57.5|56.6
`gemini-3.5-flash-lite`	low	45.0|45.6	17.5|20.3	10.0|10.6	20.0|22.7	67.5|67.5	55.0|59.5
	medium	42.5|42.9	30.0|32.7	25.0|22.7	32.5|35.1	72.5|72.5	45.0|50.2
	high	52.5|52.5	50.0|51.4	30.0|26.6	15.0|17.8	80.0|80.0	50.0|54.2
`gemini-3.1-pro-preview`	low	50.0|50.0	15.0|16.2	2.5|2.1	60.0|59.4	90.0|90.0	75.0|76.1
	medium	55.0|54.6	35.0|35.8	20.0|19.6	60.0|59.6	95.0|95.0	90.0|89.6
	high	70.0|70.0	32.5|32.8	27.5|27.1	95.0|94.6	100.0|100.0	92.5|92.7
`gemini-3.8-flash`	low	50.0|50.0	57.5|58.7	52.5|50.0	92.5|91.9	87.5|87.5	67.5|67.9
	medium	55.0|55.0	65.0|66.2	62.5|62.1	100.0|100.0	95.0|95.0	95.0|95.7
	high	57.5|57.5	85.0|86.2	90.0|90.0	100.0|100.0	97.5|97.5	95.0|95.4

Table 9: **Task-specific analysis/evaluation performance for curriculum configurations L6–L10.** Each cell reports complete-task success (%) | task-native score $\times 100$.

Model	Reasoning	Formal.	Chain	Diagnosis	Status	Semantics	Perturb.
Gemma							
`google/gemma-4-E2B-it`	off	0.0|0.0	0.0|1.8	0.0|6.4	0.0|44.6	0.0|36.2	0.0|2.6
	on	0.0|0.0	0.0|0.2	0.0|7.1	0.0|47.8	0.0|39.9	0.0|2.7
`google/gemma-4-E4B-it`	off	0.0|2.4	2.5|4.0	0.0|11.7	0.0|67.8	0.0|54.0	0.0|6.9
	on	0.0|15.2	2.5|3.8	0.0|13.2	0.0|70.7	0.0|55.6	0.0|10.5
`google/gemma-4-26B-A4B-it`	off	2.5|40.8	0.0|0.0	0.0|22.2	2.5|72.2	0.0|57.1	0.0|20.6
	on	10.0|34.9	0.0|1.2	2.5|45.8	2.5|74.6	0.0|58.5	0.0|34.9
`google/gemma-4-31B-it`	off	7.5|60.0	20.0|20.0	0.0|32.3	0.0|74.0	0.0|66.9	0.0|46.9
	on	50.0|79.9	15.0|15.0	17.5|71.5	10.0|71.8	2.5|76.0	0.0|54.0
GPT-OSS							
`openai/gpt-oss-20b`	low	0.0|7.9	0.0|5.2	0.0|9.1	0.0|35.8	0.0|36.7	0.0|0.5
	medium	5.0|50.3	40.0|45.3	0.0|20.5	0.0|69.2	0.0|50.2	0.0|13.6
	high	7.5|62.2	42.5|44.7	0.0|34.0	0.0|64.5	0.0|52.9	0.0|13.5
`openai/gpt-oss-120b`	low	5.0|26.5	30.0|30.7	0.0|14.1	0.0|66.2	0.0|38.8	0.0|8.5
	medium	5.0|57.7	50.0|52.9	0.0|34.7	0.0|74.8	0.0|58.3	0.0|22.0
	high	7.5|74.8	37.5|37.5	2.5|57.6	15.0|78.7	2.5|65.6	0.0|28.0
Qwen							
`Qwen/Qwen3.5-0.8B`	off	0.0|0.0	0.0|0.0	0.0|5.2	0.0|14.2	0.0|1.7	0.0|1.4
	on	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0
`Qwen/Qwen3.5-2B`	off	0.0|0.0	0.0|0.0	0.0|6.7	0.0|30.4	0.0|22.7	0.0|1.1
	on	0.0|0.9	0.0|0.0	0.0|1.1	0.0|1.0	0.0|7.7	0.0|0.0
`Qwen/Qwen3.5-4B`	off	0.0|22.5	22.5|28.1	0.0|33.4	0.0|78.0	0.0|55.1	0.0|24.5
	on	2.5|22.6	35.0|37.8	0.0|40.2	7.5|78.8	2.5|60.3	0.0|22.8
`Qwen/Qwen3.5-9B`	off	0.0|25.3	20.0|22.7	0.0|45.3	12.5|85.3	0.0|62.5	0.0|41.7
	on	5.0|36.2	30.0|36.2	10.0|58.7	15.0|79.2	0.0|63.2	0.0|52.4
`Qwen/Qwen3.5-35B-A3B`	off	10.0|45.9	25.0|30.3	7.5|70.2	17.5|89.3	10.0|76.9	0.0|57.8
	on	5.0|41.5	35.0|39.5	25.0|79.1	40.0|93.1	5.0|80.4	2.5|58.6
`Qwen/Qwen3.5-27B`	off	15.0|54.1	40.0|40.0	5.0|67.7	10.0|79.9	0.0|72.3	0.0|68.2
	on	17.5|54.4	37.5|65.6	35.0|86.2	25.0|86.4	0.0|75.0	2.5|75.5
`Qwen/Qwen3.8-27B`	low	27.5|71.3	32.5|32.5	22.5|73.2	2.5|69.1	0.0|42.9	5.0|43.0
	medium	30.0|81.8	35.0|39.7	32.5|80.2	5.0|69.3	0.0|63.5	0.0|50.5
	xhigh	32.5|84.6	62.5|64.9	47.5|91.8	40.0|93.2	10.0|75.4	17.5|67.0
DeepSeek							
`deepseek-v4.1-flash`	high	65.0|90.0	52.5|52.5	55.0|88.9	27.5|90.3	7.5|67.9	50.0|84.6
Gemini							
`gemini-2.5-pro`	low	45.0|57.7	25.0|34.1	0.0|25.6	0.0|70.4	0.0|52.9	0.0|25.4
	medium	32.5|59.2	35.0|44.4	37.5|89.4	35.0|93.0	20.0|74.8	20.0|92.0
	high	35.0|56.5	50.0|61.7	27.5|86.0	25.0|92.2	2.5|73.8	17.5|91.2
`gemini-3.5-flash-lite`	low	15.0|36.0	20.0|20.0	0.0|19.0	2.5|60.1	0.0|57.3	0.0|18.3
	medium	55.0|85.7	20.0|22.2	0.0|63.8	7.5|83.3	2.5|63.8	0.0|36.8
	high	82.5|86.9	37.5|38.8	30.0|82.6	20.0|89.4	2.5|71.9	2.5|67.5
`gemini-3.1-pro-preview`	low	97.5|89.1	42.5|67.3	40.0|89.7	5.0|28.3	10.0|76.5	20.0|78.9
	medium	97.5|87.2	65.0|90.1	45.0|91.5	0.0|0.0	42.5|90.4	50.0|90.6
	high	97.5|89.3	50.0|93.1	52.5|92.9	25.0|56.8	77.5|95.2	72.5|94.9
`gemini-3.8-flash`	low	95.0|89.0	62.5|68.3	50.0|92.5	60.0|96.4	60.0|95.0	57.5|86.2
	medium	97.5|94.2	85.0|93.5	50.0|92.5	62.5|97.0	87.5|97.9	95.0|99.5
	high	100.0|96.3	92.5|96.5	50.0|92.5	57.5|96.8	97.5|99.8	95.0|99.5

Table 10: **Task-specific constructive performance for curriculum configurations L6–L10.** Each cell reports complete-task success (%) | task-native score $\times 100$.

Model	Reasoning	Pref.	Counter	Counter+S	Attack	Defence	Attack–Def.
Gemma							
`google/gemma-4-E2B-it`	off	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.4	0.0|0.0	0.0|0.4
	on	0.0|0.4	0.0|0.2	0.0|0.0	0.0|0.4	0.0|0.0	0.0|0.6
`google/gemma-4-E4B-it`	off	5.0|6.8	0.0|1.4	0.0|0.9	0.0|2.6	2.5|4.5	0.0|2.1
	on	2.5|7.4	0.0|2.6	0.0|2.5	0.0|3.3	0.0|1.7	2.5|6.2
`google/gemma-4-26B-A4B-it`	off	27.5|30.9	0.0|2.1	2.5|5.8	0.0|3.6	20.0|20.7	12.5|17.9
	on	30.0|33.2	2.5|8.3	22.5|24.0	12.5|18.4	17.5|20.5	15.0|20.1
`google/gemma-4-31B-it`	off	40.0|41.8	5.0|8.9	10.0|13.6	10.0|11.8	30.0|33.2	10.0|15.5
	on	45.0|45.3	15.0|18.2	15.0|17.7	20.0|20.9	47.5|48.8	37.5|41.0
GPT-OSS							
`openai/gpt-oss-20b`	low	0.0|0.1	0.0|0.5	0.0|0.1	0.0|1.0	12.5|11.6	2.5|10.7
	medium	12.5|17.3	12.5|13.5	15.0|13.5	10.0|11.9	42.5|43.1	32.5|35.8
	high	25.0|27.8	30.0|27.7	22.5|21.8	10.0|12.6	35.0|35.9	32.5|36.3
`openai/gpt-oss-120b`	low	2.5|6.4	7.5|8.1	2.5|2.5	10.0|11.7	37.5|37.2	22.5|23.1
	medium	25.0|28.7	17.5|15.5	17.5|16.7	7.5|7.4	42.5|42.9	57.5|55.1
	high	62.5|63.3	42.5|38.9	12.5|12.3	17.5|18.0	55.0|54.3	52.5|52.4
Qwen							
`Qwen/Qwen3.5-0.8B`	off	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0
	on	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0
`Qwen/Qwen3.5-2B`	off	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.2
	on	0.0|0.0	0.0|0.0	0.0|0.2	0.0|0.0	2.5|2.5	0.0|0.0
`Qwen/Qwen3.5-4B`	off	35.0|36.4	0.0|2.7	5.0|6.8	10.0|12.6	15.0|16.0	2.5|9.1
	on	45.0|45.5	22.5|21.9	17.5|18.4	10.0|13.2	32.5|33.9	20.0|22.6
`Qwen/Qwen3.5-9B`	off	42.5|43.0	15.0|17.9	5.0|8.6	5.0|7.8	20.0|22.2	22.5|25.6
	on	45.0|45.0	30.0|30.6	32.5|33.6	17.5|19.3	35.0|37.4	45.0|45.0
`Qwen/Qwen3.5-35B-A3B`	off	40.0|41.2	7.5|10.6	10.0|12.6	5.0|7.1	22.5|24.1	15.0|20.8
	on	45.0|45.0	20.0|24.2	57.5|53.8	27.5|29.9	37.5|38.5	42.5|42.6
`Qwen/Qwen3.5-27B`	off	47.5|48.0	17.5|20.7	40.0|35.0	27.5|29.2	50.0|50.5	47.5|49.0
	on	50.0|50.0	42.5|44.5	40.0|39.8	45.0|47.2	50.0|52.3	50.0|50.8
`Qwen/Qwen3.8-27B`	low	32.5|32.8	22.5|26.6	60.0|58.6	17.5|21.7	45.0|46.8	45.0|46.1
	medium	35.0|36.1	32.5|36.3	42.5|44.4	20.0|24.7	55.0|56.6	37.5|40.5
	xhigh	50.0|50.0	47.5|48.5	15.0|15.0	42.5|38.4	42.5|44.5	57.5|56.5
DeepSeek							
`deepseek-v4.1-flash`	high	57.5|57.5	32.5|32.8	12.5|12.6	37.5|37.4	52.5|53.0	60.0|58.8
Gemini							
`gemini-2.5-pro`	low	47.5|47.7	0.0|1.6	5.0|5.2	12.5|16.2	27.5|28.7	10.0|14.6
	medium	57.5|57.5	22.5|25.7	15.0|14.6	17.5|18.9	35.0|36.8	55.0|55.1
	high	47.5|47.9	30.0|32.5	5.0|5.8	25.0|27.4	45.0|47.0	40.0|42.1
`gemini-3.5-flash-lite`	low	15.0|18.7	7.5|9.3	2.5|3.8	10.0|10.6	47.5|47.7	22.5|25.6
	medium	40.0|41.5	20.0|21.9	7.5|8.1	15.0|18.1	62.5|63.8	32.5|39.9
	high	45.0|46.0	32.5|34.0	7.5|6.9	15.0|15.4	50.0|50.3	42.5|50.2
`gemini-3.1-pro-preview`	low	50.0|50.0	55.0|55.8	20.0|17.4	32.5|33.5	72.5|73.0	70.0|71.4
	medium	50.0|50.0	67.5|68.8	25.0|23.7	30.0|29.8	60.0|61.5	90.0|88.9
	high	60.0|60.0	70.0|71.0	32.5|30.9	50.0|50.3	72.5|73.5	90.0|90.3
`gemini-3.8-flash`	low	50.0|50.0	45.0|44.7	27.5|22.9	72.5|72.6	85.0|84.9	60.0|61.6
	medium	52.5|52.5	75.0|75.2	27.5|25.1	100.0|100.0	90.0|90.0	97.5|97.2
	high	57.5|57.5	85.0|85.5	65.0|62.2	100.0|100.0	90.0|90.0	100.0|100.0

Table 11: **Task-specific analysis/evaluation performance for curriculum configurations L11–L15.** Each cell reports complete-task success (%) | task-native score $\times 100$.

Model	Reasoning	Formal.	Chain	Diagnosis	Status	Semantics	Perturb.
Gemma							
`google/gemma-4-E2B-it`	off	0.0|0.0	0.0|0.3	0.0|6.7	0.0|42.1	0.0|37.7	0.0|0.5
	on	0.0|0.0	0.0|0.0	0.0|6.7	0.0|43.4	0.0|39.5	0.0|2.5
`google/gemma-4-E4B-it`	off	0.0|15.9	0.0|0.8	0.0|8.2	0.0|59.0	0.0|50.7	0.0|3.3
	on	0.0|24.9	2.5|3.3	0.0|10.7	0.0|58.6	0.0|52.8	0.0|7.5
`google/gemma-4-26B-A4B-it`	off	0.0|39.8	0.0|2.3	0.0|10.6	0.0|66.2	0.0|61.2	0.0|10.2
	on	5.0|49.1	5.0|6.4	0.0|28.4	2.5|63.7	0.0|58.3	0.0|29.8
`google/gemma-4-31B-it`	off	7.5|68.7	7.5|8.9	0.0|26.4	5.0|72.0	0.0|65.0	0.0|36.8
	on	47.5|80.6	12.5|12.5	2.5|44.6	5.0|74.2	0.0|75.0	0.0|48.7
GPT-OSS							
`openai/gpt-oss-20b`	low	0.0|1.2	0.0|1.1	0.0|7.5	0.0|28.1	0.0|33.1	0.0|0.3
	medium	0.0|50.5	2.5|7.1	0.0|29.8	0.0|57.7	2.5|51.2	0.0|7.1
	high	0.0|61.1	17.5|21.3	0.0|28.8	0.0|55.9	0.0|55.8	0.0|8.8
`openai/gpt-oss-120b`	low	0.0|19.6	0.0|3.2	0.0|13.2	0.0|56.2	0.0|41.0	0.0|4.3
	medium	2.5|61.6	20.0|20.3	0.0|30.0	0.0|73.2	0.0|62.3	0.0|13.2
	high	2.5|75.6	27.5|27.9	0.0|44.9	2.5|75.6	0.0|66.0	0.0|24.0
Qwen							
`Qwen/Qwen3.5-0.8B`	off	0.0|0.0	0.0|0.0	0.0|3.7	0.0|6.8	0.0|4.1	0.0|0.2
	on	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0
`Qwen/Qwen3.5-2B`	off	0.0|0.0	0.0|0.7	0.0|3.0	0.0|15.1	0.0|20.4	0.0|0.3
	on	0.0|2.1	0.0|0.0	0.0|1.1	0.0|0.0	0.0|13.3	0.0|0.0
`Qwen/Qwen3.5-4B`	off	0.0|12.6	12.5|19.5	0.0|22.7	2.5|76.8	0.0|54.2	0.0|13.4
	on	2.5|7.2	27.5|32.2	2.5|32.3	5.0|60.7	0.0|60.5	0.0|13.0
`Qwen/Qwen3.5-9B`	off	0.0|23.4	15.0|19.3	0.0|33.5	7.5|84.9	0.0|59.5	0.0|28.6
	on	7.5|25.7	20.0|26.1	2.5|47.4	2.5|76.9	0.0|69.6	0.0|27.8
`Qwen/Qwen3.5-35B-A3B`	off	2.5|31.6	17.5|23.3	0.0|38.9	2.5|84.3	7.5|78.8	0.0|39.5
	on	5.0|26.3	10.0|13.5	5.0|58.4	12.5|87.6	2.5|79.6	0.0|39.1
`Qwen/Qwen3.5-27B`	off	10.0|51.2	17.5|17.5	0.0|58.4	20.0|88.8	2.5|73.3	0.0|56.0
	on	22.5|67.7	27.5|27.5	15.0|78.6	10.0|84.5	5.0|79.8	0.0|60.4
`Qwen/Qwen3.8-27B`	low	30.0|76.9	15.0|15.0	5.0|57.6	2.5|66.1	0.0|61.2	0.0|33.7
	medium	25.0|67.0	27.5|27.8	17.5|67.7	2.5|72.4	2.5|65.2	0.0|51.4
	xhigh	32.5|87.5	27.5|29.6	42.5|91.2	25.0|85.3	5.0|71.9	7.5|66.1
DeepSeek							
`deepseek-v4.1-flash`	high	52.5|93.1	37.5|37.8	37.5|83.1	22.5|88.3	2.5|66.9	40.0|81.8
Gemini							
`gemini-2.5-pro`	low	27.5|51.6	5.0|13.8	0.0|15.6	0.0|66.9	2.5|55.0	0.0|17.0
	medium	47.5|75.6	20.0|28.4	17.5|83.6	10.0|91.0	12.5|75.4	2.5|81.5
	high	32.5|67.9	20.0|29.3	20.0|83.8	12.5|90.4	12.5|76.0	2.5|87.2
`gemini-3.5-flash-lite`	low	5.0|34.4	10.0|12.1	0.0|10.8	0.0|48.8	2.5|59.8	0.0|11.7
	medium	47.5|88.7	10.0|10.6	7.5|66.8	5.0|75.8	2.5|66.7	0.0|20.0
	high	80.0|89.2	22.5|22.8	12.5|79.2	7.5|88.0	2.5|72.3	0.0|60.3
`gemini-3.1-pro-preview`	low	95.0|91.6	25.0|45.2	45.0|91.3	2.5|27.7	2.5|81.5	10.0|68.8
	medium	100.0|93.5	15.0|68.8	55.0|93.0	0.0|4.7	40.0|92.5	30.0|92.1
	high	95.0|93.5	27.5|84.4	55.0|93.3	12.5|47.5	57.5|94.6	45.0|95.6
`gemini-3.8-flash`	low	97.5|93.3	15.0|22.6	50.0|91.3	30.0|92.9	47.5|93.8	30.0|61.3
	medium	97.5|96.6	80.0|88.8	50.0|92.5	50.0|97.1	92.5|98.3	80.0|96.2
	high	97.5|98.3	65.0|86.3	50.0|92.5	50.0|97.1	97.5|99.6	82.5|98.7

Table 12: **Task-specific constructive performance for curriculum configurations L11–L15.** Each cell reports complete-task success (%) | task-native score $\times 100$.

Model	Reasoning	Pref.	Counter	Counter+S	Attack	Defence	Attack–Def.
Gemma							
`google/gemma-4-E2B-it`	off	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.1	0.0|0.0	0.0|0.1
	on	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.1	0.0|0.0	0.0|0.4
`google/gemma-4-E4B-it`	off	2.5|5.5	0.0|0.2	0.0|0.0	0.0|2.2	0.0|4.0	0.0|1.1
	on	2.5|6.0	0.0|0.8	0.0|2.4	0.0|2.3	0.0|6.3	0.0|3.5
`google/gemma-4-26B-A4B-it`	off	22.5|26.2	0.0|2.2	5.0|6.7	0.0|0.8	10.0|13.2	10.0|13.8
	on	32.5|34.3	5.0|11.0	2.5|7.3	5.0|7.6	10.0|16.9	15.0|20.5
`google/gemma-4-31B-it`	off	32.5|35.6	0.0|4.8	12.5|15.1	0.0|2.6	5.0|12.0	5.0|12.0
	on	45.0|45.8	15.0|20.0	12.5|15.0	20.0|21.7	12.5|19.4	27.5|32.9
GPT-OSS							
`openai/gpt-oss-20b`	low	0.0|0.1	0.0|0.2	2.5|2.5	0.0|0.2	0.0|1.0	10.0|14.9
	medium	10.0|11.9	12.5|12.7	10.0|10.0	2.5|4.9	17.5|18.6	17.5|21.2
	high	7.5|11.8	17.5|17.1	20.0|20.3	17.5|17.8	22.5|24.5	47.5|50.2
`openai/gpt-oss-120b`	low	0.0|2.8	2.5|3.8	0.0|0.8	0.0|1.3	12.5|13.2	17.5|19.2
	medium	32.5|35.0	5.0|6.6	7.5|8.7	2.5|5.3	27.5|29.0	37.5|36.9
	high	45.0|46.0	15.0|15.1	7.5|6.9	5.0|6.4	30.0|30.1	57.5|56.9
Qwen							
`Qwen/Qwen3.5-0.8B`	off	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.1
	on	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0
`Qwen/Qwen3.5-2B`	off	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.1
	on	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0
`Qwen/Qwen3.5-4B`	off	20.0|25.2	0.0|2.5	2.5|3.8	0.0|2.4	0.0|7.0	2.5|8.3
	on	37.5|40.0	5.0|7.3	5.0|7.3	12.5|14.1	12.5|16.4	12.5|16.4
`Qwen/Qwen3.5-9B`	off	35.0|37.4	0.0|4.5	2.5|6.2	0.0|4.3	17.5|22.2	15.0|20.4
	on	37.5|38.6	5.0|10.1	27.5|29.7	20.0|21.2	15.0|19.9	27.5|32.1
`Qwen/Qwen3.5-35B-A3B`	off	30.0|33.7	2.5|3.6	2.5|5.0	0.0|2.5	10.0|15.8	15.0|19.6
	on	45.0|46.3	10.0|13.8	25.0|28.7	22.5|22.7	30.0|35.2	37.5|40.7
`Qwen/Qwen3.5-27B`	off	47.5|47.5	10.0|12.4	7.5|8.5	15.0|15.0	25.0|29.4	45.0|47.3
	on	47.5|48.0	35.0|38.2	32.5|31.2	32.5|36.2	32.5|37.2	55.0|56.9
`Qwen/Qwen3.8-27B`	low	37.5|39.5	25.0|29.5	42.5|40.8	15.0|19.6	22.5|28.2	30.0|34.1
	medium	47.5|48.0	37.5|42.6	37.5|37.4	7.5|11.8	37.5|41.9	42.5|44.4
	xhigh	55.0|55.0	42.5|44.1	7.5|7.5	22.5|21.3	22.5|28.0	55.0|54.1
DeepSeek							
`deepseek-v4.1-flash`	high	60.0|60.2	32.5|35.0	2.5|3.4	27.5|27.7	30.0|30.2	75.0|72.3
Gemini							
`gemini-2.5-pro`	low	37.5|39.3	0.0|1.6	0.0|0.5	0.0|1.3	5.0|7.3	12.5|17.5
	medium	50.0|50.4	30.0|33.4	17.5|18.9	10.0|11.2	20.0|24.7	40.0|41.8
	high	55.0|55.0	30.0|33.1	2.5|4.7	20.0|21.1	20.0|24.2	37.5|40.5
`gemini-3.5-flash-lite`	low	0.0|4.4	2.5|4.0	0.0|0.2	5.0|5.3	20.0|21.1	10.0|16.5
	medium	42.5|43.6	17.5|20.5	0.0|1.2	7.5|8.9	47.5|48.5	35.0|40.1
	high	35.0|37.9	7.5|8.9	7.5|7.8	10.0|11.1	30.0|32.5	32.5|40.3
`gemini-3.1-pro-preview`	low	47.5|47.6	70.0|70.8	32.5|29.6	32.5|32.0	47.5|49.1	62.5|64.9
	medium	52.5|52.5	67.5|69.9	17.5|15.8	17.5|17.5	42.5|44.0	80.0|80.8
	high	55.0|55.0	70.0|72.8	10.0|9.5	17.5|18.0	57.5|59.8	92.5|93.0
`gemini-3.8-flash`	low	55.0|55.6	52.5|52.4	10.0|8.8	50.0|50.2	77.5|77.8	52.5|55.7
	medium	65.0|65.0	90.0|90.5	5.0|4.4	95.0|95.0	87.5|87.5	90.0|90.9
	high	75.0|75.0	90.0|90.3	25.0|22.5	97.5|97.5	90.0|90.0	97.5|97.8

D.4 Argument-Ordering Analysis

ArgGYM varies argument comparison along two formal dimensions: the argument comparison principle (last-link versus weakest-link) and the set ordering used to lift preferences (democratic versus elitist). These dimensions are not equally active in every generated theory. Democratic and elitist comparison can differ only when the relevant comparison involves multi-element sets. Under last-link, arguments in the current generators almost always contribute a single last defeasible rule; consequently, the democratic and elitist variants induce the same defeat relation on eleven of the twelve tasks, with `status_query` providing the designed exception. Under weakest-link, multi-element comparison is more readily available, with the set-ordering distinction becoming directly task-relevant in particular for `preference_construction`. Table 13 reports the benchmark-wide condition slices for completeness, but the democratic and elitist columns should therefore not be interpreted as a uniform experimental contrast across all tasks.

As a separate implementation diagnostic, we also recomputed each frozen theory after switching only the argument comparison principle between last-link and weakest-link while preserving the set ordering. The resulting grounded status map changed for 107 of 120 `status_query` theories and 101 of 120 `defeat_diagnosis` theories, as well as for all 120 theories in `preference_construction`, `claim_chain`, and `perturbation`. Thus, the link principle is formally active in the generated theories underlying several of the largest observed ordering differences. This same-theory diagnostic is distinct from the benchmark performance comparison below, whose last-link and weakest-link instances are independently generated and therefore remain descriptive rather than causal.

The argument comparison principle produces large benchmark-wide differences for many configurations. Gemini 3.8 Flash at high effort, for example, scores 92.9% under last-link and 71.8% under weakest-link; corresponding differences are 50.1% versus 28.5% for Qwen3.5-27B with thinking enabled and 50.7% versus 39.3% for DeepSeek-v4.1-Flash. The benchmark-wide democratic and elitist aggregates are often much closer, but those differences require a different interpretation. Because the set-ordering distinction is formally inactive in many cells, particularly under last-link, differences between the aggregate democratic and elitist columns can reflect both genuinely set-ordering-sensitive instances and variation among independently generated theories. They should therefore not be read as a benchmark-wide estimate of model sensitivity to democratic versus elitist comparison.

The aggregate direction should not be interpreted as weakest-link being uniformly harder. Task-level reversals occur. For example, DeepSeek-v4.1-Flash performs better under weakest-link on `claim_chain` (66.7% versus 53.3%) and `defeat_diagnosis` (73.3% versus 28.3%). The ordering analysis therefore describes model sensitivity to different preference regimes rather than defining a universal ordering of task difficulty.

D.4.1 Task-Specific Last-Link and Weakest-Link Results

The benchmark-wide ordering aggregates can mask substantial task-level variation. Tables 14 and 15 therefore report complete-task success separately for last-link and weakest-link within each task. Each task-specific condition contains 60 frozen benchmark instances for a complete model–reasoning configuration. Cells report *last-link (%)* | *weakest-link (%)*.

Table 13: **Performance by argument and set ordering.** Values are complete-task success (%). Last-link and weakest-link refer to the argument comparison principle; democratic and elitist refer to the set ordering used for preference comparison.

Model	Reasoning	Last-link	Weakest-link	Democratic	Elitist
Gemma					
`google/gemma-4-E2B-it`	off	1.0	1.2	1.4	0.8
	on	1.5	1.1	1.2	1.4
`google/gemma-4-E4B-it`	off	5.1	1.9	3.9	3.2
	on	4.9	2.6	3.9	3.6
`google/gemma-4-26B-A4B-it`	off	14.0	4.0	9.3	8.8
	on	20.8	7.4	14.7	13.5
`google/gemma-4-31B-it`	off	22.6	6.1	16.0	12.8
	on	37.4	17.2	27.6	26.9
GPT-OSS					
`openai/gpt-oss-20b`	low	9.9	2.6	6.5	6.0
	medium	23.3	8.8	16.4	15.7
	high	29.9	12.6	23.2	19.3
`openai/gpt-oss-120b`	low	19.6	5.1	12.1	12.6
	medium	32.4	10.7	21.5	21.5
	high	35.4	22.5	28.5	29.4
Qwen					
`Qwen/Qwen3.5-0.8B`	off	0.0	0.0	0.0	0.0
	on	0.0	0.0	0.0	0.0
`Qwen/Qwen3.5-2B`	off	0.6	0.1	0.3	0.4
	on	1.2	0.8	1.4	0.7
`Qwen/Qwen3.5-4B`	off	18.3	6.5	12.8	12.1
	on	31.1	13.6	21.5	23.2
`Qwen/Qwen3.5-9B`	off	26.1	9.6	17.9	17.8
	on	35.8	16.4	26.7	25.6
`Qwen/Qwen3.5-35B-A3B`	off	31.2	12.1	21.8	21.5
	on	42.8	20.6	30.8	32.5
`Qwen/Qwen3.5-27B`	off	42.2	20.7	30.4	32.5
	on	50.1	28.5	38.6	40.0
`Qwen/Qwen3.8-27B`	low	35.0	21.8	28.6	28.2
	medium	36.8	22.1	29.9	29.0
	xhigh	51.1	27.2	37.1	41.2
DeepSeek					
`deepseek-v4.1-flash`	high	50.7	39.3	42.8	47.2
Gemini					
`gemini-2.5-pro`	low	23.3	9.9	17.4	15.8
	medium	44.0	25.0	35.7	33.3
	high	43.6	22.8	33.5	32.9
`gemini-3.5-flash-lite`	low	22.5	13.9	16.8	19.6
	medium	37.5	21.1	28.6	30.0
	high	44.3	26.0	37.1	33.2
`gemini-3.1-pro-preview`	low	57.6	39.0	47.2	49.4
	medium	64.2	45.7	54.2	55.7
	high	70.8	55.0	63.9	61.9
`gemini-3.8-flash`	low	75.1	47.8	59.2	63.7
	medium	89.6	66.0	77.4	78.2
	high	92.9	71.8	83.2	81.5

Table 14: **Task-specific last-link versus weakest-link performance on analysis/evaluation tasks.** Each cell reports complete-task success as last-link (%) | weakest-link (%).

Model	Reasoning	Formal.	Chain	Diagnosis	Status	Semantics	Perturb.
Gemma							
google/gemma-4-E2B-it	off	0.0|1.7	0.0|5.0	5.0|5.0	0.0|3.3	0.0|0.0	0.0|0.0
	on	0.0|1.7	3.3|6.7	0.0|0.0	3.3|5.0	0.0|0.0	0.0|0.0
google/gemma-4-E4B-it	off	5.0|3.3	3.3|1.7	6.7|5.0	13.3|10.0	3.3|1.7	0.0|1.7
	on	5.0|3.3	5.0|8.3	10.0|8.3	11.7|10.0	0.0|0.0	1.7|0.0
google/gemma-4-26B-A4B-it	off	10.0|5.0	3.3|1.7	10.0|8.3	15.0|15.0	3.3|5.0	3.3|0.0
	on	16.7|5.0	16.7|15.0	10.0|11.7	11.7|10.0	11.7|8.3	1.7|3.3
google/gemma-4-31B-it	off	15.0|13.3	21.7|26.7	11.7|3.3	8.3|5.0	11.7|8.3	5.0|1.7
	on	53.3|48.3	30.0|31.7	30.0|21.7	25.0|15.0	21.7|23.3	5.0|8.3
GPT-OSS							
openai/gpt-oss-20b	low	5.0|6.7	11.7|6.7	10.0|6.7	6.7|1.7	1.7|0.0	0.0|0.0
	medium	8.3|3.3	30.0|33.3	16.7|11.7	3.3|3.3	5.0|5.0	0.0|0.0
	high	13.3|8.3	36.7|43.3	20.0|13.3	5.0|5.0	3.3|0.0	0.0|0.0
openai/gpt-oss-120b	low	10.0|5.0	28.3|26.7	10.0|5.0	6.7|5.0	1.7|5.0	0.0|0.0
	medium	13.3|5.0	46.7|51.7	13.3|8.3	20.0|10.0	10.0|11.7	0.0|0.0
	high	13.3|15.0	51.7|38.3	23.3|16.7	18.3|13.3	15.0|11.7	5.0|5.0
Qwen							
Qwen/Qwen3.5-0.8B	off	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0
	on	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0
Qwen/Qwen3.5-2B	off	1.7|1.7	1.7|0.0	1.7|0.0	0.0|0.0	0.0|0.0	0.0|0.0
	on	0.0|1.7	0.0|5.0	8.3|3.3	1.7|0.0	0.0|0.0	0.0|0.0
Qwen/Qwen3.5-4B	off	6.7|5.0	35.0|28.3	16.7|8.3	18.3|6.7	13.3|10.0	0.0|3.3
	on	11.7|3.3	50.0|50.0	25.0|15.0	28.3|13.3	8.3|8.3	0.0|5.0
Qwen/Qwen3.5-9B	off	8.3|8.3	35.0|41.7	21.7|11.7	35.0|11.7	8.3|6.7	6.7|5.0
	on	11.7|8.3	35.0|50.0	26.7|18.3	23.3|16.7	18.3|8.3	1.7|5.0
Qwen/Qwen3.8-27B	low	38.3|31.7	36.7|40.0	25.0|30.0	5.0|6.7	5.0|6.7	5.0|10.0
	medium	40.0|21.7	46.7|46.7	21.7|48.3	10.0|3.3	11.7|6.7	1.7|1.7
	xhigh	35.0|33.3	58.3|65.0	63.3|36.7	71.7|16.7	21.7|15.0	25.0|23.3
Qwen/Qwen3.5-35B-A3B	off	20.0|11.7	38.3|35.0	25.0|15.0	40.0|18.3	28.3|31.7	6.7|8.3
	on	10.0|15.0	45.0|43.3	36.7|26.7	56.7|21.7	21.7|26.7	16.7|15.0
Qwen/Qwen3.5-27B	off	20.0|16.7	48.3|43.3	23.3|16.7	38.3|13.3	23.3|20.0	8.3|6.7
	on	18.3|26.7	46.7|53.3	53.3|25.0	51.7|20.0	23.3|21.7	20.0|10.0
DeepSeek							
deepseek-v4.1-flash	high	63.3|71.7	53.3|66.7	28.3|73.3	46.7|21.7	16.7|15.0	53.3|40.0
Gemini							
gemini-2.5-pro	low	43.3|40.0	28.3|20.0	16.7|11.7	6.7|6.7	1.7|1.7	0.0|1.7
	medium	43.3|45.0	48.3|50.0	45.0|26.7	48.3|21.7	28.3|25.0	23.3|26.7
	high	43.3|36.7	43.3|55.0	50.0|26.7	53.3|21.7	25.0|18.3	21.7|23.3
gemini-3.5-flash-lite	low	21.7|21.7	28.3|36.7	18.3|15.0	20.0|13.3	10.0|11.7	0.0|1.7
	medium	60.0|56.7	45.0|31.7	31.7|13.3	26.7|20.0	18.3|10.0	0.0|6.7
	high	80.0|88.3	56.7|43.3	53.3|16.7	35.0|16.7	20.0|18.3	10.0|10.0
gemini-3.1-pro-preview	low	95.0|96.7	53.3|51.7	85.0|18.3	31.7|15.0	30.0|28.3	33.3|31.7
	medium	98.3|100.0	58.3|56.7	95.0|18.3	26.7|15.0	60.0|53.3	56.7|43.3
	high	95.0|100.0	68.3|48.3	100.0|21.7	46.7|18.3	75.0|78.3	66.7|61.7
gemini-3.8-flash	low	95.0|100.0	56.7|60.0	96.7|18.3	85.0|25.0	70.0|65.0	55.0|51.7
	medium	96.7|100.0	88.3|88.3	100.0|15.0	100.0|26.7	90.0|95.0	96.7|71.7
	high	98.3|100.0	80.0|91.7	100.0|15.0	96.7|26.7	98.3|98.3	95.0|78.3

Table 15: **Task-specific last-link versus weakest-link performance on constructive tasks.** Each cell reports complete-task success as last-link (%) | weakest-link (%).

Model	Reasoning	Pref.	Counter	Counter+S	Attack	Defence	Attack–Def.
Gemma							
`google/gemma-4-E2B-it`	off	5.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	1.7|0.0	0.0|0.0
	on	6.7|0.0	0.0|0.0	0.0|0.0	0.0|0.0	5.0|0.0	0.0|0.0
`google/gemma-4-E4B-it`	off	25.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	5.0|0.0	0.0|0.0
	on	16.7|0.0	0.0|0.0	0.0|0.0	0.0|0.0	8.3|0.0	0.0|1.7
`google/gemma-4-26B-A4B-it`	off	61.7|0.0	3.3|1.7	3.3|5.0	0.0|0.0	35.0|6.7	20.0|0.0
	on	70.0|0.0	10.0|6.7	23.3|8.3	8.3|10.0	40.0|5.0	30.0|5.0
`google/gemma-4-31B-it`	off	81.7|0.0	10.0|0.0	21.7|8.3	15.0|1.7	53.3|3.3	16.7|1.7
	on	93.3|0.0	18.3|10.0	26.7|6.7	25.0|15.0	56.7|21.7	63.3|5.0
GPT-OSS							
`openai/gpt-oss-20b`	low	16.7|0.0	5.0|1.7	5.0|0.0	5.0|3.3	28.3|3.3	23.3|1.7
	medium	36.7|0.0	31.7|6.7	15.0|8.3	18.3|13.3	66.7|10.0	48.3|10.0
	high	46.7|0.0	40.0|16.7	36.7|11.7	28.3|8.3	60.0|20.0	68.3|25.0
`openai/gpt-oss-120b`	low	26.7|0.0	21.7|0.0	8.3|5.0	18.3|6.7	56.7|3.3	46.7|0.0
	medium	68.3|0.0	25.0|6.7	23.3|3.3	21.7|5.0	73.3|10.0	73.3|16.7
	high	88.3|25.0	23.3|38.3	21.7|16.7	16.7|10.0	65.0|55.0	83.3|25.0
Qwen							
`Qwen/Qwen3.5-0.8B`	off	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0
	on	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0
`Qwen/Qwen3.5-2B`	off	1.7|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0	0.0|0.0
	on	3.3|0.0	0.0|0.0	0.0|0.0	0.0|0.0	1.7|0.0	0.0|0.0
`Qwen/Qwen3.5-4B`	off	63.3|0.0	6.7|0.0	8.3|3.3	8.3|10.0	33.3|1.7	10.0|1.7
	on	86.7|0.0	33.3|10.0	20.0|23.3	20.0|15.0	53.3|6.7	36.7|13.3
`Qwen/Qwen3.5-9B`	off	81.7|0.0	16.7|5.0	13.3|5.0	6.7|5.0	45.0|5.0	35.0|10.0
	on	86.7|0.0	31.7|10.0	50.0|38.3	28.3|20.0	53.3|11.7	63.3|10.0
`Qwen/Qwen3.8-27B`	low	65.0|1.7	26.7|25.0	56.7|45.0	26.7|16.7	53.3|38.3	76.7|10.0
	medium	71.7|6.7	31.7|23.3	40.0|41.7	23.3|10.0	73.3|41.7	70.0|13.3
	xhigh	91.7|8.3	63.3|21.7	21.7|1.7	36.7|18.3	53.3|38.3	71.7|48.3
`Qwen/Qwen3.5-35B-A3B`	off	78.3|1.7	16.7|5.0	25.0|6.7	11.7|0.0	48.3|5.0	36.7|6.7
	on	93.3|0.0	23.3|8.3	50.0|35.0	41.7|20.0	60.0|15.0	58.3|20.0
`Qwen/Qwen3.5-27B`	off	93.3|1.7	26.7|20.0	31.7|31.7	43.3|31.7	68.3|25.0	81.7|21.7
	on	98.3|0.0	38.3|45.0	48.3|36.7	41.7|46.7	71.7|31.7	90.0|25.0
DeepSeek							
`deepseek-v4.1-flash`	high	100.0|20.0	33.3|35.0	13.3|13.3	35.0|30.0	66.7|41.7	98.3|43.3
Gemini							
`gemini-2.5-pro`	low	83.3|6.7	5.0|1.7	5.0|5.0	13.3|16.7	53.3|5.0	23.3|1.7
	medium	88.3|18.3	23.3|23.3	23.3|8.3	16.7|13.3	58.3|21.7	81.7|20.0
	high	95.0|10.0	26.7|25.0	6.7|1.7	20.0|20.0	60.0|23.3	78.3|11.7
`gemini-3.5-flash-lite`	low	40.0|0.0	16.7|1.7	5.0|3.3	16.7|6.7	60.0|30.0	33.3|25.0
	medium	83.3|0.0	25.0|20.0	13.3|8.3	21.7|15.0	75.0|46.7	50.0|25.0
	high	86.7|1.7	33.3|26.7	18.3|11.7	10.0|16.7	68.3|38.3	60.0|23.3
`gemini-3.1-pro-preview`	low	98.3|0.0	45.0|48.3	13.3|23.3	40.0|43.3	81.7|58.3	85.0|53.3
	medium	100.0|5.0	48.3|65.0	30.0|11.7	33.3|38.3	68.3|63.3	95.0|78.3
	high	100.0|23.3	48.3|66.7	25.0|21.7	58.3|50.0	66.7|86.7	100.0|83.3
`gemini-3.8-flash`	low	100.0|3.3	53.3|50.0	33.3|26.7	71.7|71.7	90.0|76.7	95.0|25.0
	medium	100.0|15.0	80.0|73.3	40.0|23.3	98.3|98.3	85.0|96.7	100.0|88.3
	high	100.0|26.7	95.0|78.3	61.7|58.3	100.0|98.3	90.0|95.0	100.0|95.0

D.5 Generation and Answer-Protocol Diagnostics

Canonical complete-task success remains the benchmark score throughout the paper. We additionally track generation truncation and whether a response contains the required `<answer>` region. To quantify the extent to which missing answer regions affect a run, Table 16 also reports success conditioned on the presence of an answer region. This conditioned value is diagnostic only: it divides canonical successes by responses containing an answer region and does not recover or rescore malformed outputs.

Most configurations have high answer-region compliance. The two smallest Qwen3.5 thinking-on configurations are notable exceptions: Qwen3.5-0.8B and Qwen3.5-2B truncate 99.2% and 82.6% of generations and achieve answer-region compliance of 32.4% and 28.7%, respectively. Their near-zero canonical success therefore reflects both task performance and severe generation/protocol effects. Across the remaining configurations, conditioning on compliant responses changes success only modestly. The largest differences occur for Gemini 2.5 Pro, where conditioned success exceeds canonical success by 3.2–3.9 percentage points; Qwen3.8-27B at low effort shows a 2.0-point difference, and Qwen3.5-4B with thinking enabled a 1.9-point difference. The principal benchmark conclusions are therefore not explained by answer-protocol failures alone.

There are no scorer refusals or API errors in the final scored runs. Canonical success is retained throughout because producing a valid submitted answer is part of the benchmark contract.

Table 16: **Generation and answer-protocol diagnostics.** Trunc. is the percentage of truncated generations; No answer is the percentage without the required <answer> region; Compliance is the corresponding answer-region compliance rate. Cond. denotes success conditioned on the presence of an answer region, and Δ is conditioned minus canonical success in percentage points.

Model	Reasoning	Trunc.	No answer	Compliance	Canonical	Cond.	Δ
Gemma							
google/gemma-4-E2B-it	off	0.4	0.8	99.2	1.1	1.1	+0.0
	on	0.1	20.9	79.1	1.3	1.7	+0.3
google/gemma-4-E4B-it	off	0.0	0.2	99.8	3.5	3.5	+0.0
	on	0.1	4.2	95.8	3.8	3.9	+0.2
google/gemma-4-26B-A4B-it	off	10.3	9.9	90.1	9.0	10.0	+1.0
	on	1.9	2.0	98.0	14.1	14.4	+0.3
google/gemma-4-31B-it	off	1.7	2.0	98.0	14.4	14.7	+0.3
	on	0.0	0.1	99.9	27.3	27.3	+0.0
GPT-OSS							
openai/gpt-oss-20b	low	0.0	3.6	96.4	6.2	6.5	+0.2
	medium	0.0	4.0	96.0	16.0	16.7	+0.7
	high	0.4	1.1	98.9	21.2	21.5	+0.2
openai/gpt-oss-120b	low	0.0	0.4	99.6	12.4	12.4	+0.1
	medium	0.0	0.1	99.9	21.5	21.6	+0.0
	high	0.0	0.4	99.7	29.0	29.1	+0.1
Qwen							
Qwen/Qwen3.5-0.8B	off	4.7	15.9	84.1	0.0	0.0	+0.0
	on	99.2	67.6	32.4	0.0	0.0	+0.0
Qwen/Qwen3.5-2B	off	1.7	16.2	83.8	0.3	0.4	+0.1
	on	82.6	71.3	28.7	1.0	3.6	+2.6
Qwen/Qwen3.5-4B	off	0.1	0.4	99.7	12.4	12.5	+0.0
	on	8.3	7.8	92.2	22.4	24.3	+1.9
Qwen/Qwen3.5-9B	off	0.4	0.2	99.8	17.8	17.9	+0.0
	on	1.2	4.3	95.7	26.1	27.3	+1.2
Qwen/Qwen3.5-35B-A3B	off	4.4	0.7	99.3	21.7	21.8	+0.2
	on	3.3	2.8	97.2	31.7	32.6	+0.9
Qwen/Qwen3.5-27B	off	0.1	0.1	99.9	31.5	31.5	+0.0
	on	0.6	1.4	98.6	39.3	39.9	+0.6
Qwen/Qwen3.8-27B	low	0.0	6.5	93.5	28.4	30.4	+2.0
	medium	0.0	1.5	98.5	29.4	29.9	+0.5
	xhigh	0.2	0.2	99.8	39.2	39.2	+0.1
DeepSeek							
deepseek-v4.1-flash	high	0.0	0.0	100.0	45.0	45.0	+0.0
Gemini							
gemini-2.5-pro	low	0.0	16.0	84.0	16.6	19.8	+3.2
	medium	0.0	10.2	89.8	34.5	38.4	+3.9
	high	0.0	9.9	90.1	33.2	36.9	+3.7
gemini-3.5-flash-lite	low	0.0	0.3	99.7	18.2	18.2	+0.1
	medium	0.0	0.0	100.0	29.3	29.3	+0.0
	high	0.0	0.4	99.6	35.1	35.3	+0.1
gemini-3.1-pro-preview	low	0.0	0.3	99.7	48.3	48.5	+0.1
	medium	0.0	0.4	99.6	54.9	55.2	+0.2
	high	0.0	0.5	99.5	62.9	63.2	+0.3
gemini-3.8-flash	low	0.0	0.8	99.2	61.5	61.9	+0.5
	medium	0.0	0.1	99.9	77.8	77.9	+0.1
	high	0.0	0.0	100.0	82.4	82.4	+0.0