REASONABLE-RLVR: STEP-VERIFIED REWARDS FOR REASONING OVER KNOWLEDGE GRAPHS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with verifiable rewards (RLVR) trains reasoning models with programs that check their outputs. Such programs check answers in mathematics and code, and they check steps in formal theorem proving. Knowledge-intensive domains such as medicine have no such language, so a correct answer reached by unsound reasoning receives full credit. Knowledge graphs make entailment decidable, but no reward checks cited steps against their closure.

We introduce Reasonable-RLVR, which verifies each cited proof step against the OWL 2 RL closure of a graph and checks whether the steps establish the answer. The verifier tests the supported chain against a relation contract, has configurable coverage, and needs no reference proof or learned judge. We define two versions of the reward, a two-band form that satisfies five ordering properties on anchored traces and a simpler weighted sum.

Reasonable-RLVR obtains a higher correct-and-valid rate than outcome and REFERENCE-PATH rewards on four knowledge-graph benchmarks. Without a supervised warm-up on WebQSP, it reaches 0.597 ± 0.120 correct-and-valid proofs over three seeds, while the outcome reward reaches 0.000 at 94% answer accuracy. Verifier feedback at inference also improves frontier-model proofs. These results extend verifiable rewards to reasoning over knowledge graphs.

1 INTRODUCTION

Reinforcement learning from verifiable rewards scores sampled outputs with a program and updates the policy toward higher scores (Guo et al., 2025; Shao et al., 2024). It needs no learned reward model and no labels beyond the answer key, and usually checks only the final answer. For a trace $\tau = (z_1, \ldots, z_T, \hat{y})$, the outcome reward $R_{\text{out}}(\tau) = \mathbf{1}[\hat{y} = y]$ checks no intermediate step.

This approach applies when answers can be matched to a reference or programs tested against a test suite (Chen et al., 2021). Formal theorem proving also permits step verification through a language with a checker (Kim & Yun, 2026). In fields such as medicine, an answer may be a diagnosis, drug or test. An answer key checks correctness but cannot distinguish sound from unsound reasoning.

Knowledge graphs allow programs to check reasoning against encoded facts and rules. A claim is supported if it appears in the graph or follows from those facts and rules, so verification depends on the graph's available knowledge. §2 illustrates this with a medical graph linking aortic root dilation to Marfan syndrome and echocardiography.

Proofs of the same answer may use different findings, redundant support or unrelated facts. Outcome rewards score them equally, while supervised fine-tuning on one reference trace treats alternatives as mismatches. A verifier resolves citations, checks whether the proof establishes the answer, and identifies unnecessary steps. We call this *explicit process verification*. It differs from answer matching, query execution, reference-path overlap (Yan et al., 2026b), and language-model judgments of step plausibility (Yuan et al., 2026b). Figs. 1 and 2 show the loop and three proofs of one answer.

In this work we study whether a process reward can train models to produce correct answers and valid proofs without reference proofs. We compare it with answer- and path-based rewards and test whether reduced verifier coverage directs learning toward unchecked steps. Knowledge graphs allow coverage to be reduced by masking specified edges, so the effect of each omission can be measured. Across the four benchmarks, our reward outperforms outcome and REFERENCE-PATH

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

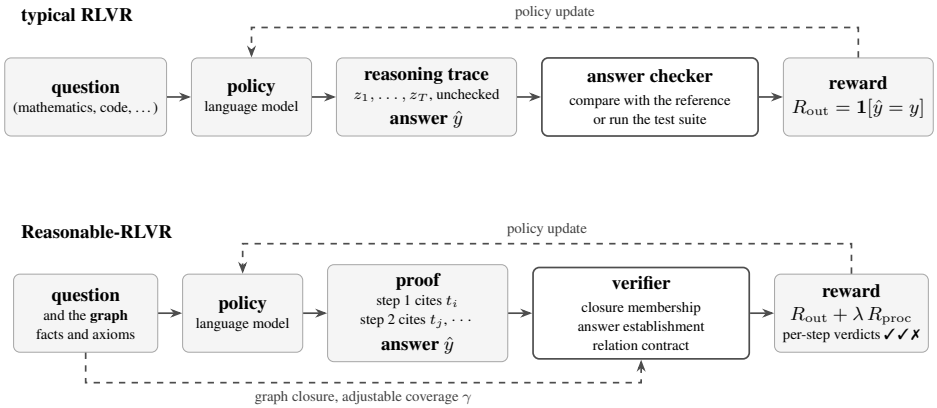


Figure 1: Outcome-only RLVR (top) and Reasonable-RLVR (below), which rewards verified steps.

baselines and ties the strongest published shape using the same verifier. It also learns valid proofs where outcome reward learns none. Reduced coverage lowers proof validity on small graphs, where abstention on unseen steps restores it.

Contributions.

1. A verifier that checks cited proof steps against OWL 2 RL closure, tests the supported chain against a relation contract, and provides controllable coverage across nine graphs.
2. A verifier-based reward that needs no reference proof and outperforms REFERENCE-PATH and outcome rewards, with five ordering properties that a two-band variant meets.
3. A framework for comparing reward designs under one verifier with controllable coverage.
4. Experiments on four real graphs showing gains over outcome and REFERENCE-PATH rewards, learning without warm-up, sensitivity to coverage, and inference-time proof repair.
5. Nine knowledge graphs with proof contracts, to be released with the harness.

2 Question Answering with Supporting Subgraph

We consider question answering with a supporting subgraph (Zhang et al., 2025). The model receives a question and triples with fact identifiers covering the reference path and a capped neighbourhood. The subgraph therefore contains the answer, simulates perfect retrieval, and separates reasoning from retrieval. The model writes one cited proof step per line, followed by an answer.

In Fig. 2, the question asks which test monitors the condition indicated by a patient's finding. Citing t1, t2 and t3 connects the patient to echocardiography through hasFinding, indicates and monitoredBy, and each citation resolves to a supplied fact. Other valid traces use a different finding, cite both findings, or first mention the patient's medication. Outcome rewards score all four equally. The verifier accepts them and marks unused supported steps, which the reward may penalise.

A trace is *fully valid* when every cited claim is supported by the graph's closure and the supported chain establishes the stated answer under the question's contract. Our primary metric, the *correct-and-valid rate*, is the fraction of evaluation questions answered correctly with a fully valid trace. We report the *fully valid trace rate* and answer accuracy separately because a valid trace may state an answer that differs from the reference. Unless stated otherwise, n counts evaluation questions.

3 Reasonable-RLVR

3.1 Reasonable-RLVR Verifier

The verifier checks proof steps against the supporting graph's OWL 2 RL closure (Motik et al., 2012). Forward chaining is decidable because the rules introduce no new names and the closure is finite. We materialise and cache it once per graph using *reasonable*, a Rust OWL 2 RL reasoner,

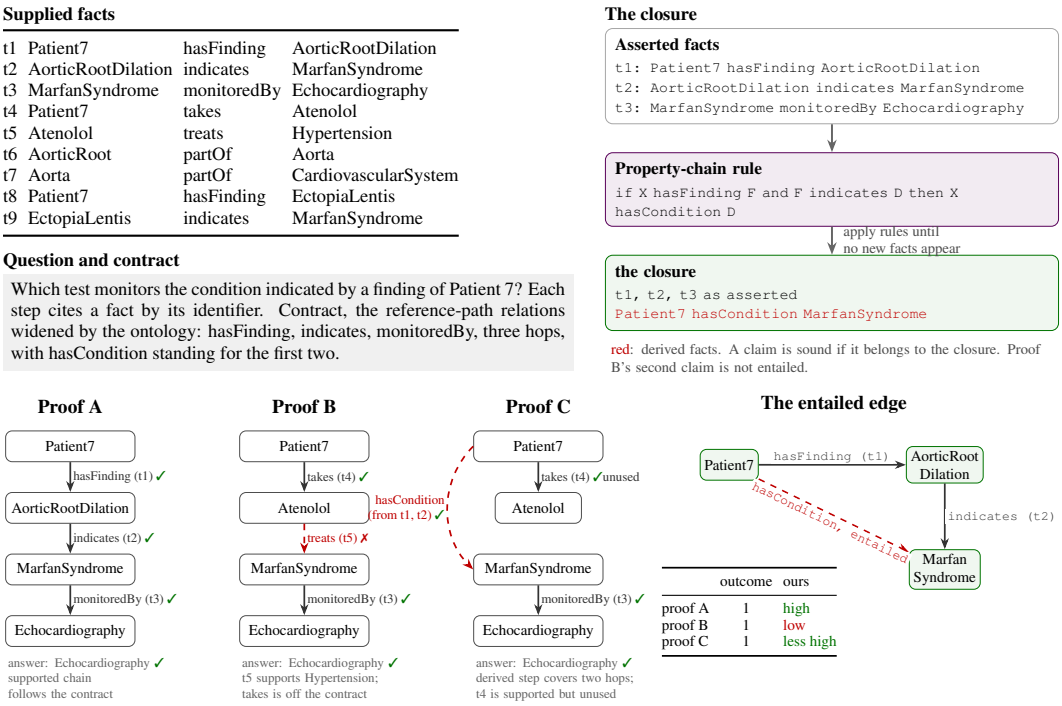


Figure 2: Three proofs of the same answer. A uses asserted facts, B includes an unsupported step, and C uses a derived fact and an unused supported step. A and C are sound.

with an auxiliary rule layer for property chains. Each claim line is parsed into a triple, its identifier is resolved to a graph fact, and soundness is checked by closure membership. Claims outside the closure are rejected even when true in the world (App. B).

For each trace, the verifier records parsing success, citation resolution, soundness, chain connectivity, answer establishment, contradictions, fabricated identifiers and unused sound steps. A connected chain starts at the question's entity and links each step's object to the next step's subject. Establishment also requires the chain to reach the stated answer through the question's relation sequence. The question's *contract* specifies the required relation chain, its hop count and, where the graph declares types, the answer type. It is the relation sequence of the question's reference path, widened by the ontology to sub- and super-properties, inverses and transitive runs. It specifies the question as a test suite specifies a programming problem: the verifier reads it, the policy never sees it, and any chain of entailed steps that realises it is credited. The entailment check is shared by all nine graphs, and 2Wiki comparison questions use a separate comparison contract (App. B). Fig. 2 illustrates the closure, contract and step verdicts for three proofs. These verdicts supply the verifier-based rewards, including the re-implemented baselines in §5.

Masking closure edges reduces verifier coverage, allowing us to measure the effect of steps that cannot be confirmed. A strict verifier rejects unseen claims. An abstaining verifier gives them neither credit nor penalty. The verifier accepts all reference proofs on the four benchmarks and agrees with an independent checker using external reasoners on 97.8 to 99.8% of traces (App. B).

3.2 Explicit Process Reward

The reward combines answer correctness and proof validity, so a correct answer alone does not earn full credit. The outcome term uses the answer, and the process term uses the verifier's verdicts. For a trace τ with stated answer $\hat{y}$ and reference answer y,

$$R(\tau) = R_{\text{out}}(\tau) + \lambda\, R_{\text{proc}}(\tau), \qquad R_{\text{out}}(\tau) = \mathbf{1}[\hat{y} = y], \tag{1}$$

with $\lambda = 1$ in every run. A trace is *parsed* if it contains at least one well-formed claim line, its answer is the last answer line or, failing that, its last line, and it is *fabricated* if it cites an identifier

Table 1: Compared rewards. Signal gives the process input and step check. Ref. gives the reference proof or path information the process reward reads, excluding the answer key, zero supports learning without proof labels, and cov. allows coverage control. Arm and reads describe our implementation. App. G lists exclusions.

work	signal (check)	ref.	zero	cov.	arm	reads
controls (Guo et al., 2025; Shao et al., 2024)	the answer	no	yes	no	OUTCOME-ONLY	the answer, the RLVR standard
					OUTCOME WITH FORMAT	the answer, given a parsed claim line
K2V (Yuan et al., 2026b)	LM judge over a checklist (judged)	no	yes	no	GATED	answer + answer × our process term
					K2V-JUDGE	language-model judge, our criteria
Explore-on-Graph (Yan et al., 2026b)	reference-path recall (overlap)	yes	no	no	REFERENCE-PATH	answer + reference-path recall
					EOG-RELEASE	recall only, released code
VPRM (Pronesti et al., 2026)	step labels (lookup)	yes	no	no	STEP-SUM	answer + share of supported claims
Ours	**verifier verdicts relations (entailment)**		**yes**	**yes**	**CDU**	$R_{\text{out}} + R_{\text{proc}}$**, §3.2**
					COU	**two-band form, App. A**

that resolves to no fact in the graph. The verifier reports *soundness* $s \in [0, 1]$, the fraction of supported claims, whose citations resolve, whose rule licenses the conclusion and whose conclusion is in the closure, and *progress* $p \in [0, 1]$, the fraction of the question's h required hops covered by the supported chain from its entity. It also reports $\mathbf{1}_{\text{est}}$ when that chain reaches $\hat{y}$ through the required relations, $\mathbf{1}_{\text{contra}}$ when a claim contradicts the closure, and $\mathbf{1}_{s=1}$ when every claim is supported. For a parsed, non-fabricated trace,

$$R_{\text{proc}}(\tau) = \min\Big(1,\ \max\big(0,\ w_b + w_s\, s + w_p\, p - w_c\, \mathbf{1}_{\text{contra}}\big) + w_{\text{sb}}\, \mathbf{1}_{s=1} + w_e\, \mathbf{1}_{\text{est}}\Big), \qquad (2)$$

while an unparsed trace receives $R_{\text{proc}} = 0$ and a fabricated trace receives $R_{\text{proc}} = c_{\text{fab}}$. The positive weights reward parsing (w_b), support (w_s), progress (w_p), full soundness (w_{sb}) and establishment (w_e), while w_c penalises contradiction. They sum to one, keeping the process term in $[0, 1]$ on the same scale as the outcome term. Establishment has the largest weight, placing complete proofs above partial ones. Progress comes next and rewards each covered hop. Small contradiction and fabrication penalties place such traces below otherwise identical clean ones. §5 gives the values.

We call this the *combined dense ungated* reward (CDU). *Combined* denotes the sum of outcome and process rewards. *Dense* refers to progress credit for each covered hop, which encourages longer supported chains. *Ungated* means that establishment credit is awarded whenever the supported chain reaches the answer through the required relations, whatever the other claims. A finished proof with one unsupported claim thus outranks a clean partial attempt. The *combined* variant omits progress, moves its weight to support and establishment, and credits establishment only for fully sound traces.

Properties. Over the process rewards enumerated in App. A, each credited hop increases CDU's reward, an establishing trace outranks every non-establishing trace, and an unsupported claim never increases reward. With the weights in §5, establishing traces score at least 0.65 in the process term and others at most 0.60 (Proposition 1). App. A states two further properties that CDU does not satisfy: penalising unused supported claims and rewarding the final hop at least as much as any earlier hop. The two-band *combined ordered ungated* reward (COU) satisfies all five on anchored traces over the same feature space. §6 evaluates it as the ordered reward.

4 Related work

Process and formal verification. Process rewards come from learned reward models (Uesato et al., 2022; Lightman et al., 2024; Sullivan & Koller, 2026; Palandye et al., 2026), prompted judges (Yuan et al., 2026b) or formal step checkers (Kim & Yun, 2026; Xu et al., 2026; Chen et al., 2026). A judge's error rate is not known without an audit, and formal checkers exclude natural-language proofs (App. G). Audits at evaluation time count correct answers with invalid proofs (Zheng et al., 2026; Helff et al., 2026).

Knowledge-graph process rewards. Explore-on-Graph rewards reference-path recall (Yan et al., 2026b). K2V gates a language-model judge's checklist score on answer correctness, and the judge does not see the graph (Yuan et al., 2026b). VPRM uses reference step labels in a medical task

(Pronesti et al., 2026). §5 defines our implementations and distinguishes reward-shape comparisons from the original systems. Other methods use reference chains, lexical path support or corpus co-occurrence (Kansal & Jha, 2026; Sun & Kazakov, 2026; Li et al., 2026; Fan et al., 2026). App. G discusses their supervision and why they are excluded from the experiments. None checks cited steps against the graph closure.

Verifier quality and padding. Studies of imperfect verifiers vary answer-level error, scalar-score accuracy or a synthetic oracle (Plesner et al., 2026; Egashira et al., 2026; Yuan et al., 2026a). With incomplete evidence, a hard rule cannot avoid both false accepts and false rejects (Li & Song, 2026). Masking the closure specifies which steps go unchecked and measures their effect. Published padding results concern learned rewards (Shihab & Afrin, 2026). Verify-then-repair loops (Wang et al., 2026; Ding et al., 2026b; Zhao et al., 2026) carry the verifier's feedback at inference.

Summary. CDU and COU are the only rewards in Table 1 that check cited steps against the graph closure, test the chain against a relation contract, need no reference trace, path overlap or judge, and allow control of verifier coverage. The table compares signals, supervision requirements and verifier coverage, and names each implementation. App. G gives the full discussion.

5 Experimental setup

Data. We use ComplexWebQuestions (CWQ, 2,829 reachable evaluation questions), Dynamic-KGQA (3,765), WebQSP (1,552) and 2WikiMultihopQA (778), with supporting subgraphs from Freebase and Wikidata, together with UMLS, CoDEx-S and the synthetic graphs, nine graphs in all (App. C). Each question's evidence holds the reference path and its neighbourhood, capped at 60 triples (the full released subgraph on Dynamic-KGQA), so the answer is always present.

Training and inference. We fully fine-tune Qwen2.5-1.5B-Instruct for fixed step budgets of about three epochs with GRPO (TRL 1.8.0), no KL term and $\lambda = 1$, with advantages centred within prompt groups and divided by the batch reward standard deviation. CWQ and Dynamic-KGQA runs start from a supervised warm-up on 2,679 2Wiki reference proofs, and WebQSP and 2Wiki runs from the Instruct model. *Without labels* means no proof warm-up and no reference proof or step label, while the outcome reward reads the reference answer and the contract fixes the reference path's relation sequence (§3.1). Tables report three seeds where available and one otherwise, with greedy evaluation (App. C). At inference, Gemini 3.6 Flash, Claude Sonnet 5 and untrained Qwen2.5 models from 0.5B to 7B receive a greedy turn and up to two verifier-feedback turns on CWQ, WebQSP and Dynamic-KGQA.

Rewards. CDU (§3.2) uses $w_b = 0.10$, $w_s = 0.15$, $w_p = 0.25$, $w_{\text{sb}} = 0.10$, $w_e = 0.40$, $w_c = 0.10$ and $c_{\text{fab}} = 0.05$. COMBINED uses $w_s = 0.30$, $w_e = 0.50$ and no progress term, with other weights unchanged, and App. A gives the ordered weights. Table 1 lists the compared rewards and what each reads. The published shapes are reproduced with our verifier's verdicts, so they compare shapes rather than the original systems.

6 Results

Main comparison. Table 2 compares ten rewards on four benchmarks under batch scaling. CDU or the GATED obtains the highest correct-and-valid rate on every benchmark, and where both learn their means differ by less than one standard deviation (CWQ .704 against .708, Dynamic-KGQA .856 against .844). Answer-based rewards reach similar or higher answer accuracy with far lower correct-and-valid rates, and the gap between the two columns is the share of correct answers with invalid proofs. With supervised warm-up, OUTCOME-ONLY proves about half as many questions as CDU while answering slightly more (CWQ .374 against .704 correct-and-valid, .879 against .856 accuracy). Without warm-up it answers nearly every question and proves none (WebQSP .000 against .597, at accuracy .943). On 2Wiki, CDU is the only reward with a substantial rate on every seed ($.278 \pm .015$). COMBINED averages .132, reaching .397 on one seed of three, and every other reward stays below .03. EOG-RELEASE, run on one seed per benchmark, assigns zero answer weight in its released code, so every run copies evidence and scores zero correct-and-valid.

Table 2: Batch-scaled benchmark comparison. cav.: correct-and-valid rate, acc.: answer accuracy, CoT-P@8/pass@8: corresponding pass rates over eight samples. Cells show seed means with the SD beneath, and superscripts give counts below three seeds. Best cav. is bold, second best underlined, and ties within one SD bold and starred.

	CWQ (n=2829)				Dynamic-KGQA (n=3765)				WebQSP (n=1552)				2Wiki (n=778)			
Reward	cav.	acc.	CoT-P@8	pass@8	cav.	acc.	CoT-P@8	pass@8	cav.	acc.	CoT-P@8	pass@8	cav.	acc.	CoT-P@8	pass@8
GATED REFERENCE	**.708*** ±.024	.859 ±.006	.817	.909	.844 ±.017	.917 ±.009	.917	.952	**.624*** ±.047	.930 ±.004	.748	.956	.000 ±.000	.775 ±.024	.000	.833
K2V JUDGE	.240 ±.116	.865 ±.003	.341	.903	.528 ±.194	.913 ±.011	.692	.956	.000 ±.000	.929 ±.012	.000	.963	.000 ±.001	.829 ±.027	.000	.899
EOG PATH	.485 ±.052	.864 ±.005	.759	.912	.721 ±.019	.911 ±.007	.900	.957	.000 ±.000	.931 ±.004	.000	.961	.000 ±.000	.796 ±.030	.000	.905
EOG RELEASE	.000$^{\dagger 1}$	.000$^{\dagger 1}$	.001	.001	.000$^{\dagger 1}$	.000$^{\dagger 1}$	.001	.002	.000$^{\dagger 1}$	.001$^{\dagger 1}$	.000	.048	.000$^{\dagger 1}$	.000$^{\dagger 1}$	.000	.005
STEP SUM	.403 ±.028	.868 ±.007	.487	.897	.506 ±.044	.918 ±.003	.554	.943	.044 ±.016	.931 ±.002	.100	.950	.028 ±.028	.768 ±.016	.072	.836
OUTCOME ONLY	.374 ±.056	.879 ±.006	.503	.915	.401 ±.352	.922 ±.004	.473	.947	.000 ±.000	.943 ±.004	.000	.955	.000 ±.000	.810 ±.005	.000	.867
OUTCOME + FMT	.401 ±.009	.882 ±.004	.533	.913	.576 ±.045	.917 ±.006	.656	.946	.026 ±.026	.940 ±.004	.084	.963	.001 ±.001	.795 ±.025	.018	.894
CDU (OURS)	**.704*** ±.010	.856 ±.014	.814	.904	**.856*** ±.012	.926 ±.005	.928	.956	**.597*** ±.120	.922 ±.005	.729	.950	**.278*** ±.015	.758 ±.014	.457	.827
COMBINED (OURS)	.685 ±.032	.855 ±.003	.791	.903	**.851*** ±.012	.918 ±.005	.919	.952	.556 ±.116	.921 ±.013	.621	.942	**.132*** ±.229	.771 ±.024	.198	.842
COU	.603 ±.022	.847 ±.010	.660	.878	.747 ±.020	.922 ±.008	.793	.946	.542 ±.076	.910 ±.005	.596	.934	.000 ±.000	.791 ±.018	.000	.840

Sampling eight traces per question raises the correct-and-valid rate of every reward that is above zero at one sample but does not close the gap. With warm-up, OUTCOME-ONLY at eight samples stays below CDU at one (CWQ .503 against .704, Dynamic-KGQA .473 against .856). Without warm-up it stays at zero while its answer pass rate reaches .955 on WebQSP and .867 on 2Wiki (Table 19).

Failure modes. Table 14 (App. D) reports the share of questions answered correctly with invalid proofs, single-claim share and local validity. The last measures whether all claims are supported, distinguishing unsupported proofs from sound chains that stop short of the answer. Without warm-up, OUTCOME-ONLY answers .94 of WebQSP questions and writes no claim lines, and answers .81 on 2Wiki with almost no locally valid traces. Verifier-based rewards can instead collapse to one-claim traces. This occurs on one CDU seed and two COMBINED seeds on WebQSP, and every GATED and COU seed on 2Wiki. In non-collapsed runs, about nine in ten traces are locally valid even where the correct-and-valid rate is low, so most remaining failures contain only supported claims yet do not establish the correct answer. CDU also produces 5.1 to 7.5 unused supported claims per valid proof, compared with 0.3 to 1.3 for COU (Table 7), and the REFERENCE-PATH reward pads more than CDU on CWQ and Dynamic-KGQA (App. D, Table 15). COU reduces padding but achieves lower correct-and-valid rates and collapses more often (App. D).

Small and synthetic graphs. Experiments on UMLS, CoDEx-S and synthetic graphs show the same advantage of verifier-based rewards without proof labels, with exceptions at 0.5B and at partial label budgets, and a collapse without warm-up that varies by seed and graph. Without reference proofs, the policy reaches 0.38 on held-out four-hop synthetic questions. The coverage finding differs: abstention recovers losses on the small synthetic graphs but not on Dynamic-KGQA. App. F gives these experiments and their earlier protocols.

Summary. Across the four benchmarks, step verification against the graph obtains higher correct-and-valid rates than answer-based and REFERENCE-PATH rewards and matches the strongest published shape, but remains exposed to collapse without warm-up and to padding.

6.1 ABLATIONS

Model size and repair. The trained 1.5B model outperforms the untrained 7B model in correct-and-valid rate. With warm-up, it reaches .70 to .86, close to the frontier models' zero-shot rates of .72 to .94 (Fig. 3). Two verifier-feedback turns on rejected traces improve every model of 3B or larger. The 7B model gains .10 to .13 and Sonnet .07 to .13. Trained policies improve modestly without lower shipped-answer accuracy. Tables 3 and 5 give the per-model and trained-policy results. Training against the verifier therefore narrows the gap in correct-and-valid rate between the 1.5B model and the evaluated frontier models, and within a two-turn repair loop the same verifier raises the validity of models it never trained.

	CWQ		WebQSP		Dynamic-KGQA	
model	t0	t2	t0	t2	t0	t2
Qwen 0.5B	.006	.006	.006	.006	.033	.041
Qwen 1.5B	.009	.010	.074	.079	.034	.035
Qwen 3B	.104	.140	.246	.322	.198	.243
Qwen 7B	.143	.240	.298	.425	.291	.404
Gemini 3.6 Flash	.901	.935	.940	.975	.845	.983
Claude Sonnet 5	.724	.857	.826	.940	.852	.921

Table 3: Correct-and-valid rate of untrained models before feedback (t0) and after two feedback turns (t2), one seed per model.

Setting	n	warm-up only	RL, CDU	RL, COU
CWQ, 6 epochs	2829	.338 ± .010	.704 ± .010	.603 ± .022
Dynamic-KGQA, 6 epochs	3765	.347 ± .012	.856 ± .012	.747 ± .020
WebQSP, 0 epochs	1552	$.000^{\dagger 1}$	$.463^{\dagger 1}$	$.497^{\dagger 1}$
WebQSP, 1 epoch	1552	$.498^{\dagger 1}$	$.754^{\dagger 1}$	$.760^{\dagger 1}$
WebQSP, 2 epochs	1552	$.492^{\dagger 1}$	$.846^{\dagger 1}$	$.793^{\dagger 1}$
WebQSP, 3 epochs	1552	$.537^{\dagger 1}$	$.836^{\dagger 1}$	$.773^{\dagger 1}$
WebQSP, 6 epochs	1552	$.530^{\dagger 1}$	$.832^{\dagger 1}$	$.763^{\dagger 1}$

Table 4: Correct-and-valid rate after warm-up alone and with RL. WebQSP rows use seed 1.

	CWQ				WebQSP				Dynamic-KGQA			
	cav.		shipped acc.		cav.		shipped acc.		cav.		shipped acc.	
reward	t0	t2	t0	t2	t0	t2	t0	t2	t0	t2	t0	t2
CDU (OURS)	.707	.747	.843	.849	.633	.685	.926	.932	.841	.869	.921	.924
GATED REF.	.684	.709	.857	.857	.576	.604	.927	.928	.824	.847	.917	.921

Table 5: Trained 1.5B policies, seed 0, before feedback (t0) and after two feedback turns (t2). The shipped answer is defined in §6.1.

	correct-and-valid			fully valid			accuracy		
verifier	s0	s1	s2	s0	s1	s2	s0	s1	s2
strict	.833	.838	.844	.862	.866	.867	.911	.915	.919
abstaining	.842	.825	.842	.868	.856	.868	.916	.904	.908

Table 6: Dynamic-KGQA training with hidden reverse relations under strict or abstaining verification, evaluated with the full verifier.

	CWQ		Dynamic-KGQA		WebQSP	
reward	unused / trace	share with any	unused / trace	share with any	unused / trace	share with any
CDU (OURS)	7.47 ± 2.83	1.000 ± 0.000	6.76 ± 0.10	1.000 ± 0.000	5.12 ± 4.50	.667 ± 0.577
COMBINED (OURS)	4.08 ± 1.65	.990 ± 0.016	6.18 ± 2.31	1.000 ± 0.000	1.65 ± 2.86	.333 ± 0.577
COU	0.37 ± 0.08	.319 ± 0.062	1.31 ± 0.10	.860 ± 0.054	0.30 ± 0.52	.124 ± 0.215

Table 7: Unused supported claims per fully valid trace and the share of valid traces with at least one, mean and standard deviation over seeds.

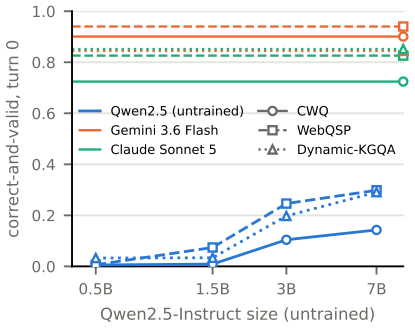


Figure 3: Zero-shot correct-and-valid rate against model size.

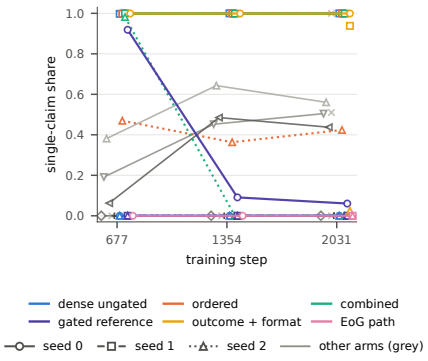


Figure 4: Single-claim share during WebQSP training without warm-up.

Warm-up. Warm-up teaches the proof form and stabilises RL. Alone it reaches .338 on CWQ, .347 on Dynamic-KGQA and .498 on WebQSP, and RL raises these to .704, .856 and .754 (Table 4). Without warm-up, WebQSP RL reaches .597 ± .120, one seed collapses to one-claim traces from the first scored checkpoint, and the healthy seeds stay below the warm-up runs (Fig. 4). With any tested warm-up no seed collapses, and the RL rate peaks after two warm-up passes on the tested seed. One run without the one-hop questions does not collapse either, at .372 against .696 over fewer steps.

Reward shape and advantage scaling. Exposure to collapse depends on the reward shape and on the advantage normalisation. COU (App. A) satisfies all five ordering properties yet trails CDU on every benchmark, .603 against .704 on CWQ and .000 against .278 on 2Wiki, and collapses

Table 8: WebQSP advantage scaling. cav.: correct-and-valid rate (mean ± seed SD), 1-cl.: share with at most one claim, dead: tied-reward group share, entr.: token entropy. Dead-group share and entropy average the final third of training. Per-seed signals appear in Table 18.

	batch scaling				group scaling			
reward	cav.	1-cl.	dead	entr.	cav.	1-cl.	dead	entr.
GATED	.624 ± .047	.020	.069	.028	.588 ± .113	.001	.659	.031
EOG PATH	.000 ± .000	.000	.157	.111	.000 ± .000	.000	.680	.119
STEP SUM	.044 ± .016	.000	.648	.037	.041 ± .008	1.000	.978	.014
OUTCOME	.000 ± .000	1.000	.798	.023	.014 ± .025	.524	.968	.078
OUTCOME + FMT	.026 ± .026	.655	.718	.133	.032 ± .014	.500	.973	.081
CDU (OURS)	.597 ± .120	.333	.033	.022	.661 ± .010	.000	.576	.024
COMBINED (OURS)	.556 ± .116	.667	.056	.021	.636 ± .051	.118	.731	.024
COU	.542 ± .076	.808	.043	.014	.475 ± .013	.989	.877	.009

most often. Its cost for a wrong claim is an unbounded count, whereas the bounded fraction of the sums stops rewarding deletion once the remaining claims are correct. Group scaling divides each completion's centred reward by the standard deviation of its prompt group, while batch scaling uses the batch reward standard deviation. Groups with identical rewards carry no gradient, and group scaling can amplify small differences within near-tied groups. In the WebQSP runs it is associated with a higher final share of tied groups for every reward (Table 8), which is consistent with reduced output diversity, although the experiment does not isolate the mechanism. COU collapses on every seed, whereas CDU and GATED do not, and under batch scaling collapse varies by seed. GATED leads CDU under batch scaling (.624 against .597), while CDU leads under group scaling (.661 against .588). CWQ favours batch scaling for every reward but OUTCOME WITH FORMAT, while on Dynamic-KGQA CDU, COMBINED and GATED change little. §7 examines the collapse in detail, and App. D, Tables 8, 16 and 17, gives the full comparisons and training signals.

Checker coverage. A verifier does not see every true fact, so we test whether penalising the correct steps it cannot confirm harms training and whether abstention, which gives unseen steps neither credit nor penalty, recovers the loss. On the small synthetic graphs both hold. Masking a quarter of the relations lowers the correct-and-valid rate from .697 to .288 at 0.5B ($n = 22$) and from .712 to .520 at 1.5B ($n = 59$), and abstention restores .803 and .723. On the larger synthetic graph masking does not lower the rate, and on UMLS abstention recovers little (App. F, Table 21). On Dynamic-KGQA we hide the declarations that let two relations be read backwards, so a third of the training questions need a reversal the verifier cannot confirm (App. C, Table 13). Over three seeds ($n = 3{,}765$) the strict verifier reaches .838 ± .006 and the abstaining one .836 ± .010, against .856 ± .012 with the full verifier (Table 2), so hiding the reversals lowers the mean by .018 and abstention does not recover it (Table 6).

7 Analysis and discussion

The one-claim collapse phenomenon. Without warm-up, some policies reduce their proofs to one claim and remain there. We classify a run as collapsed if its single-claim share stays at or above .8 over the final tenth of training, and as a dip if it reaches .8 earlier and then falls (Fig. 5). The initial policy produces eight to fourteen claims per trace, of which .62 to .86 are wrong. Every logged run shortens its traces during the first fifty steps, and the runs that collapse fall to one claim at a median step of 110 and do not recover. Seven of the runs in Fig. 5 reach .8 and later fall, one of them shown to step 1,573, so a temporary reduction to one claim does not determine the final state.

Three pressures towards collapse. Three factors favour shorter traces early in training. First, when most claims are wrong, deleting them can increase reward more than extending the supported chain. Recomputing both changes from logged rewards favours truncation on .90 to .97 of CDU traces with wrong answers and mixed supported and wrong claims, and on .97 to .99 under the ordered reward. COU penalises an unbounded wrong-claim count and the sums a bounded fraction.

Second, the answer term continues to reward one-claim policies. Collapsed 2Wiki policies answer .80 to .85 of questions correctly, although no question can be proved with one claim. Third, 68% of WebQSP questions have one-hop answers, making one-claim proofs complete on much of the

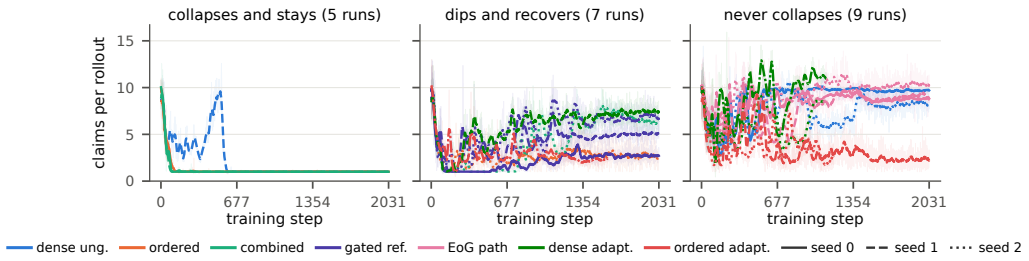


Figure 5: Claims per rollout on WebQSP without warm-up, grouped by collapse pattern as defined in §7. Faint lines show per-step values and bold lines 25-step means.

training set. Two WebQSP seeds reach the same reward of 1.80, one collapsed and one not. Training reward thus does not separate the two states, but the held-out rate does.

Once completions become similar, the token entropy falls, to .000 at the final logged step on the collapsed WebQSP seed against .024 for GATED on the same seed, and tied-reward groups carry no gradient under group scaling, which can prevent recovery. Susceptibility differs by reward. On the final greedy traces, with collapse at a .9 single-claim share, the ordered rewards collapse in 17 of 21 runs, COMBINED in 4 of 8, GATED in 2 of 8 and CDU in 1 of 8. No warm-up run collapses.

Adaptive rewards and padding. To counter the wrong-claim pressure we tried adaptive rewards, which weaken the wrong-claim penalty while most claims in recent batches are wrong (App. E), reducing the incentive to delete claims early in training. On WebQSP without warm-up, neither adaptive variant collapses at the final checkpoint across three seeds. ADAPTIVE DENSE reaches .666, compared with .597 for CDU and .624 for GATED, and ADAPTIVE ORDERED reaches .612, compared with .542 for COU. Gains concentrate on previously collapsed seeds, and seed ranges narrow (App. E, Fig. 8). The intervention has limits. ADAPTIVE ORDERED collapses on 2Wiki in the single-seed test, which is consistent with collapse sustained by the answer term, although further experiments are needed (App. E). Padding is the opposite pressure. Rewards that do not penalise unused supported claims let proofs grow, and CDU produces 5.1 to 7.5 such claims per valid proof (Table 7). COU penalises off-chain claims and pads less than CDU, COMBINED and GATED but is most susceptible to collapse, and ADAPTIVE DENSE continues to pad. A remaining design target is a reward that penalises off-chain claims (P4, App. A) without an unbounded count, and future work could also condition answer credit on proof completion or vary the process weight during training.

Limitations and future work. The benchmarks are shallow. Under our shortest-path derivation, 68% of WebQSP, 47% of CWQ, 46% of Dynamic-KGQA and 17% of 2Wiki questions have one-hop answers, only Dynamic-KGQA has a sizeable share (14%) at three or more hops, and the evidence contains the answer by construction, so the results say little about deep reasoning. The primary comparison uses one policy size, 1.5B, with supplementary 0.5B runs, and the collapse analysis is mostly limited to 1.5B. The tasks use one proof format for question answering over knowledge graphs, and the verifier has not been tested on other task families. Supplied evidence excludes retrieval errors, and questions without a contract are excluded.

8 CONCLUSION

Reasonable-RLVR extends verifiable rewards to proof verification over knowledge graphs. Its verifier checks cited steps against OWL 2 RL closure, tests the supported chain against the question's relation contract, and reports soundness and answer establishment. These verdicts support training without reference proofs or a learned judge. Across four benchmarks, CDU achieves a higher correct-and-valid rate than outcome and REFERENCE-PATH rewards, matches the strongest published shape under the same verifier, and learns valid proofs without proof labels where outcome reward learns none. Verifier feedback at inference also improves frontier models' proof validity. Future work should evaluate larger models, broader tasks, retrieved evidence and questions without contracts.

Reproducibility statement

The supplementary material contains a minimal scorer for the checker and the reward shapes over supplied pre-materialised examples, with stored policy outputs on which it reproduces the full system's verdicts. App. B specifies the verifier and App. C gives the data, splits and hyperparameters. We plan to release the verifier, rewards, nine graphs and their contracts, and training and evaluation code upon publication.

Ethics statement

We use public knowledge graphs and question-answering benchmarks, including public medical ontologies. No patient data or human subjects are involved.

AI use statement

We used generative AI to implement methods and analysis code, design experiments, generate synthetic datasets through code, assist in formulating mathematical claims, create figures, edit text and summarise literature. We did not use it to formulate research questions or hypotheses. The authors reviewed the assisted work, spot-checked verifier verdicts against the closure and an independent checker, and approved every experiment. Committed scripts extracted reported numbers from run files. We take responsibility for all final text, claims and artifacts, including AI-assisted content.

References

Aakriti Agrawal, Minghui Liu, and Furong Huang. VeriGate: Verifier-gated step-level supervision for GRPO. *arXiv preprint arXiv:2605.30451*, 2026. URL https://arxiv.org/abs/2605.30451.

Jiayue Cao, Zhicong Lu, Xuehan Sun, Wei Jia, Hongling Zheng, Changyuan Tian, Zichuan Lin, Wenqian Lv, and Nayu Liu. CORA: Analyzing and bridging thinking-answer gap in multimodal RLVR via consistency-oriented reasoning alignment. *arXiv preprint arXiv:2606.14691*, 2026. URL https://arxiv.org/abs/2606.14691.

Luoxin Chen, Yichi Zhou, and Huishuai Zhang. Learning to generate formally verifiable step-by-step logic reasoning via structured formal intermediaries. *arXiv preprint arXiv:2603.29500*, 2026. URL https://arxiv.org/abs/2603.29500.

Mark Chen et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

Ruiyi Ding, Yongxuan Lv, Xianhui Meng, Jiahe Song, Chao Wang, Chen Jiang, and Yuan Cheng. PRPO: Aligning process reward with outcome reward in policy optimization. *arXiv preprint arXiv:2601.07182*, 2026a. URL https://arxiv.org/abs/2601.07182.

Weiwei Ding, Zixuan Li, Long Bai, Zhuo Chen, Kun Su, Fei Wang, Xiaolong Jin, Jin Zhang, Jiafeng Guo, and Xueqi Cheng. Code-on-graph: Iterative programmatic reasoning via large language models on knowledge graphs. *arXiv preprint arXiv:2606.03705*, 2026b. URL https://arxiv.org/abs/2606.03705.

Kazuki Egashira, Mark Vero, Jasper Dekoninck, Florian E. Dorner, Robin Staab, and Martin Vechev. Delay, plateau, or collapse: Evaluating the impact of systematic verification error on rlvr. In *Conference on Language Modeling (COLM)*, 2026. URL https://arxiv.org/abs/2605.02909.

Shicheng Fan, Haochang Hao, Dehai Min, Weihao Liu, Philip S. Yu, and Lu Cheng. Verifiable rewards beyond math and code: Lightweight corpus-grounded process supervision for factual question answering. *arXiv preprint arXiv:2605.29648*, 2026. URL https://arxiv.org/abs/2605.29648.

Daya Guo, Dejian Yang, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025. doi:10.1038/s41586-025-09422-z.

Tianyang Han, Hengyu Shi, Junjie Hu, Xu Yang, Zhiling Wang, and Junhao Su. Correct is not enough: Training reasoning planners with executor-grounded rewards. *arXiv preprint arXiv:2605.03862*, 2026. URL https://arxiv.org/abs/2605.03862.

Lukas Helff, Quentin Delfosse, David Steinmann, Ruben Härle, Hikaru Shindo, Patrick Schramowski, Wolfgang Stammer, Kristian Kersting, and Felix Friedrich. LLMs gaming verifiers: RLVR can lead to reward hacking. ICLR 2026 Workshop on Logical Reasoning of Large Language Models, 2026. URL https://openreview.net/forum?id=g6sqdWLzV0. arXiv:2604.15149.

Yuelyu Ji, Zhuochun Li, Hui Ji, and Daqing He. StepGap: A hybrid NLI-LLM checker for step-level evidence-gap detection in multi-hop question answering. *arXiv preprint arXiv:2605.24733*, 2026. URL https://arxiv.org/abs/2605.24733.

Jinghan Jia, Joe Benton, and Eric Easley. Faithfulness as information flow: Evaluating and training faithful chain-of-thought reasoning. *arXiv preprint arXiv:2605.24286*, 2026. URL https://arxiv.org/abs/2605.24286.

Yuval Kansal and Niraj K. Jha. Knowledge graphs are implicit reward models: Path-derived signals enable compositional reasoning. *arXiv preprint arXiv:2601.15160*, 2026. URL https://arxiv.org/abs/2601.15160.

Minsu Kim and Se-Young Yun. Process-verified reinforcement learning for theorem proving via Lean. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=P00k4DFaXF. arXiv:2606.20068.

Bojie Li and Noah Shi. RLVP: Penalize the path, reward the outcome. *arXiv preprint arXiv:2607.07435*, 2026. URL https://arxiv.org/abs/2607.07435.

Jiaqi Li and Fanghui Song. Grounding LLM reasoning under incomplete graph evidence. *arXiv preprint arXiv:2606.30247*, 2026. URL https://arxiv.org/abs/2606.30247.

Zhaoyang Li, Zenghuang Fu, Qiuyuan Ai, Ping Jiang, Haoyu Wu, Minghui Wu, Chenxu Zhao, Jie Song, and Guannan He. CoEvoKG: Co-evolving knowledge graphs with self-evolving search agents. *arXiv preprint arXiv:2608.01904*, 2026. URL https://arxiv.org/abs/2608.01904.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi. arXiv:2305.20050.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=qTj3TqJN93. arXiv:2310.01061.

Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz. OWL 2 web ontology language profiles (second edition). W3C Recommendation, 11 December 2012, 2012. URL https://www.w3.org/TR/owl2-profiles/.

Anagha Radhakrishna Palandye, Rebecca Glick, and Osheen Kaul. Reward granularity in RLVR: Comparing process and outcome reward structures for mathematical reasoning in small language models. *arXiv preprint arXiv:2607.02869*, 2026. URL https://arxiv.org/abs/2607.02869.

Andreas Plesner, Francisco Guzmán, and Anish Athalye. An imperfect verifier is good enough: Learning with noisy rewards. *arXiv preprint arXiv:2604.07666*, 2026. URL https://arxiv.org/abs/2604.07666.

Massimiliano Pronesti, Anya Belz, and Yufang Hou. Beyond outcome verification: Verifiable process reward models for structured reasoning. In *Findings of the Association for Computational Linguistics: ACL 2026*, 2026. doi:10.18653/v1/2026.findings-acl.1611. arXiv:2601.17223.

Ali Rad, Khashayar Filom, Darioush Keivan, Peyman Mohajerin Esfahani, and Ehsan Kamalinejad. Rate or fate? RLV$^{\varepsilon}$R: Reinforcement learning with verifiable noisy rewards. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, PMLR 306, 2026. URL https://icml.cc/virtual/2026/poster/64587. arXiv:2601.04411.

Amirul Rahman and Mohammed Sabih Alsharari. VeriBound: PAC-Bayesian generalization bounds for process reward models trained with formal verification tools. *arXiv preprint arXiv:2606.20740*, 2026. URL https://arxiv.org/abs/2606.20740.

Jaideep Ray. Before the model learns the bug: Fuzzing RLVR verifiers. *arXiv preprint arXiv:2606.01066*, 2026. URL https://arxiv.org/abs/2606.01066.

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. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Ibne Farabi Shihab and Fariya Afrin. Quality-diversity stress tests for process reward models: What archive coverage can and cannot certify. *arXiv preprint arXiv:2608.08008*, 2026. URL https://arxiv.org/abs/2608.08008.

Michael Sullivan and Alexander Koller. GRPO is secretly a process reward model. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026. URL https://icml.cc/virtual/2026/poster/61734. arXiv:2509.21154.

Jia Ao Sun, Hao Yu, Fengran Mo, Zhan Su, Yuchen Hui, Bang Liu, and Jian-Yun Nie. Search-on-graph-R1: Training large language models to search knowledge graphs with reinforcement learning. *arXiv preprint arXiv:2607.18481*, 2026. URL https://arxiv.org/abs/2607.18481.

Tianda Sun and Dimitar Kazakov. Peak-then-collapse and the four interface channels of knowledge-graph tool use. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2026. URL https://arxiv.org/abs/2605.26037.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022. URL https://arxiv.org/abs/2211.14275.

Shuai Wang and Yinan Yu. KG-Hopper: Empowering compact open LLMs with knowledge graph reasoning via reinforcement learning. In *International Joint Conference on Neural Networks (IJCNN)*, 2026. URL https://arxiv.org/abs/2603.21440.

Zehua Wang, Zhaojin Zhang, Boyu Qiu, Xiaolong Weng, Ying Xiong, Buzhou Tang, and Min Zhang. SR-RAG: Verifiable multi-hop reasoning via on-the-fly symbolic graph construction. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 38588–38606, 2026. doi:10.18653/v1/2026.findings-acl.1922.

Wuzhenghong Wen, Chao Xue, Su Pan, Yuwei Sun, and Minlong Peng. Reinforcement learning enhanced multi-hop reasoning for temporal knowledge question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 33881–33889, 2026a. doi:10.1609/aaai.v40i40.40680. arXiv:2601.01195.

Xumeng Wen, Zihan Liu, Shun Zheng, Shengyu Ye, Zhirong Wu, Yang Wang, Zhijian Xu, Xiao Liang, Junjie Li, Ziming Miao, Jiang Bian, and Mao Yang. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base LLMs. In *International Conference on Learning Representations (ICLR)*, 2026b. URL https://openreview.net/forum?id=jGbRWwIidy. arXiv:2506.14245.

Bin Xie, Bingbing Xu, Xueyun Tian, Yilin Chen, and Huawei Shen. Towards robust process reward modeling via noise-aware learning. *arXiv preprint arXiv:2601.12748*, 2026. URL https://arxiv.org/abs/2601.12748.

Esther Xin. Where the verifier fails: A category-level audit of reward signals in RLVR. *arXiv preprint arXiv:2609.01354*, 2026. URL https://arxiv.org/abs/2609.01354.

Jundong Xu, Hao Fei, Huichi Zhou, Xin Quan, Qijun Huang, Shengqiong Wu, William Yang Wang, Mong-Li Lee, and Wynne Hsu. LogicReward: Incentivizing LLM reasoning via step-wise logical supervision. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=IRhYVOKFe0. arXiv:2512.18196.

Lecheng Yan, Ruizhe Li, Guanhua Chen, Qing Li, Jiahui Geng, Wenxi Li, Longyue Wang, and Chenyang Lyu. Spurious rewards paradox: Mechanistically understanding how RLVR activates memorization shortcuts in LLMs. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026a. URL https://icml.cc/virtual/2026/poster/63951. arXiv:2601.11061.

Shiqi Yan, Yubo Chen, Ruiqi Zhou, Zhengxi Yao, Shuai Chen, Tianyi Zhang, Shijie Zhang, Wei-Qiang Zhang, Yongfeng Huang, Haixin Duan, and Yunqi Zhang. Explore-on-graph: Incentivizing autonomous exploration of large language models on knowledge graphs with path-refined reward modeling. In *International Conference on Learning Representations (ICLR)*, 2026b. URL https://openreview.net/forum?id=NfuBj8jleE. arXiv:2602.21728.

Haotong Yang, Zitong Wang, Shijia Kang, Siqi Yang, Wenkai Yu, Xu Niu, Yike Sun, Yi Hu, Zhouchen Lin, and Muhan Zhang. ProofVerifier: A scalable, diversity-driven framework for natural-language proof verification. *arXiv preprint arXiv:2602.02377*, 2026. URL https://arxiv.org/abs/2602.02377.

Huining Yuan, Zelai Xu, Huaijie Wang, Xiangmin Yi, Jiaxuan Gao, Xiao-Ping Zhang, Yu Wang, Chao Yu, and Yi Wu. Verifiable process rewards for agentic reasoning. *arXiv preprint arXiv:2605.10325*, 2026a. URL https://arxiv.org/abs/2605.10325.

Zhonghang Yuan, Zhefan Wang, Fang Hu, Zihong Chen, Jinzhe Li, Gang Li, Jie Ying, Huanjun Kong, Songyang Zhang, and Nanqing Dong. Knowledge-to-verification: Exploring RLVR for LLMs in knowledge-intensive domains. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2026b. doi:10.18653/v1/2026.acl-long.1891. arXiv:2605.18261.

Chuyifei Zhang. When the reward suite is leaky: A preregistered causal contrast of natural verifier false positives in RLVR. *arXiv preprint arXiv:2607.11022*, 2026. URL https://arxiv.org/abs/2607.11022.

Liangliang Zhang, Zhuorui Jiang, Hongliang Chi, Haoyang Chen, Mohammed ElKoumy, Fali Wang, Qiong Wu, Zhengyi Zhou, Shirui Pan, Suhang Wang, and Yao Ma. Diagnosing and addressing pitfalls in KG-RAG datasets: Toward more reliable benchmarking. In *Advances in Neural Information Processing Systems 38 (NeurIPS), Datasets and Benchmarks Track*, pp. 75490–75527, 2025. doi:10.52202/085713-2274. arXiv:2505.23495.

Yimo Zhao, Tianyuan Hu, and Shuai Jiang. Exploiting verification asymmetry for failure-aware graph reasoning in knowledge graph question answering with llms. *Symmetry*, 18(8):1340, 2026. doi:10.3390/sym18081340.

Xinyi Zheng, Ningke Li, Xiaokun Luan, Kailong Wang, Ling Shi, Meng Sun, and Haoyu Wang. Beyond correctness: Exposing LLM-generated logical flaws in reasoning via multi-step automated theorem proving. In *Proceedings of the 48th IEEE/ACM International Conference on Software Engineering (ICSE)*, pp. 3001–3013, 2026. doi:10.1145/3744916.3787820. arXiv:2512.23511.

Chenyu Zhou, Qiliang Jiang, and Xu Zhou. Multilingual verifier bias in RLVR: Benchmark, rollout diagnosis, and the cross-lingual selection bottleneck. *arXiv preprint arXiv:2608.20362*, 2026a. URL https://arxiv.org/abs/2608.20362.

Yefan Zhou, Yilun Zhou, Austin Xu, Soroush Vosoughi, Shafiq Joty, and Jiang Gui. The hidden signal of verifier strictness: Controlling and improving step-wise verification via selective latent steering. *arXiv preprint arXiv:2605.20745*, 2026b. URL https://arxiv.org/abs/2605.20745.

Yuxuan Zhu and Daniel Kang. Noisy data is destructive to reinforcement learning with verifiable rewards. *arXiv preprint arXiv:2603.16140*, 2026. URL https://arxiv.org/abs/2603.16140.

A ORDERING PROPERTIES AND THE ORDERED REWARD

This appendix defines the five ordering properties in §3.2, the ordered reward that satisfies them, and the proposition cited in the main text. Let h be the number of hops required by the question's contract and m the number of claims in a trace. A trace is *anchored* when its first supported claim has the question's entity as subject. It is *establishing* when the supported chain reaches the stated answer through the required relations. Full validity (§2) additionally requires every claim to be supported.

The properties. For parsed traces of one question, P1 requires each additional credited hop to strictly increase reward. P2 requires an unsupported claim to strictly decrease it. P3 requires an establishing trace to outrank every non-establishing trace at any depth and length. P4 requires a supported off-chain claim to strictly decrease reward, with a larger decrease at greater distance from the chain. P5 requires the final hop to receive at least as much credit as any earlier hop. P3 concerns establishment, which the rewards use. A completed proof with an unsupported off-chain claim establishes its answer but is not fully valid. P3 requires it to outrank every non-establishing trace. Table 9 evaluates these properties over non-fabricated verifier feature tuples with a correct answer, depth 1 to 5, off-chain distance 1 to 3, anchoring and flags. It reports each property's worst margin for dense ungated, gated dense, combined and ordered rewards. The properties are verified on this enumerated domain only. Dense ungated satisfies P1 on anchored chains and P3 strictly, and P2 weakly, since an unsupported claim never increases its reward and strictly decreases it once any claim is supported. It fails P4 and P5. Combined satisfies none strictly. Ordered satisfies all five on anchored chains.

Table 9: Ordering properties over 40,240 non-fabricated states. Checks and crosses indicate whether a property holds, and values give the worst process-reward margin. P1–P4 require strict inequalities, so ties fail. Graded P4 requires increasing off-chain cost with distance.

property	DENSE UNGATED	GATED DENSE	COMBINED	ORDERED	ORDERED ANY-HOP UNGATED
P1 (anchored)	✓ +0.050	✓ +0.050	✗ 0.000	✓ +0.0079	✓ +0.0079
P1 (unanchored)	✗ 0.000	✗ 0.000	✗ 0.000	✗ 0.000	✓ +0.0088
P2	✗ 0.000	✗ 0.000	✗ 0.000	✓ +0.0046	✓ +0.0046
P3	✓ +0.080	✗ -0.320	✗ -0.440	✓ +0.381	✓ +0.381
P4	✗ -0.075	✗ -0.075	✗ -0.150	✓ +0.0006	✓ +0.0006
P4 graded	✗ 0.000	✗ 0.000	✗ 0.000	✓ +0.0006	✓ +0.0006
P5	✗ -0.070	✗ -0.070	✗ -0.140	✓ 0.000	✓ 0.000

The ordered reward. The ordered reward has two bands,

$$R_{\text{ord}} = E\,\mathbf{1}_{\text{est}} + (1-E)\,q, \qquad q = 1 - \frac{u}{1+a\,u}, \qquad u = 1 - x,$$

where x is a quality score over the verifier's features,

$$x = q_b + q_s\,(1-\omega) + q_e\,(1-\delta) + q_p\,p - q_c\,\mathbf{1}_{\text{contra}},$$

where ω is the number of unsupported claims per required hop, δ is the distance-graded cost of supported off-chain claims per required hop, and p is anchored progress. Each supported claim off the credited chain costs $\min(d,3)/3$, where d is 1 when an endpoint lies on the chain, 2 when it is one hop away in the graph of supported claims and 3 when it is disconnected, and δ is the sum of these costs divided by h. Unparsed traces score zero, and traces citing fabricated identifiers receive the cap c_{fab}. Every ordered run uses $E = 0.60$, $q_b = 0.25$, $q_s = 0.25$, $q_e = 0.10$, $q_p = 0.40$, $q_c = 0.20$, $a = 1.20$ and $c_{\text{fab}} = 0.05$. The tables report the ungated form, called ordered, which enters the upper band on establishment alone. The gated form requires a fully sound establishing trace. The transformation q keeps the lower band increasing, convex and bounded in $(1 - 1/a, 1]$, so it neither flattens nor crosses the upper band. Implemented rewards are rounded to six decimals, and the margins in Table 9 are computed before rounding.

Proposition 1. *For the process term of parsed, non-fabricated traces, under the gated dense sum a finished proof with one unsupported claim among m establishes its answer and scores at most $0.35 + 0.15(1 - 1/m) < 0.50$, while a sound trace that does not establish the answer and covers k of h hops scores $0.35 + 0.25\,k/h$, which exceeds it once $k/h \geq 0.6$. The gated sum therefore fails P3. Without the gate, every establishing trace scores at least 0.65 and every other trace at most 0.60, and P3 holds.*

The enumeration confirms the proposition, with worst margins of -0.320 for the gated reward and $+0.080$ for the ungated reward.

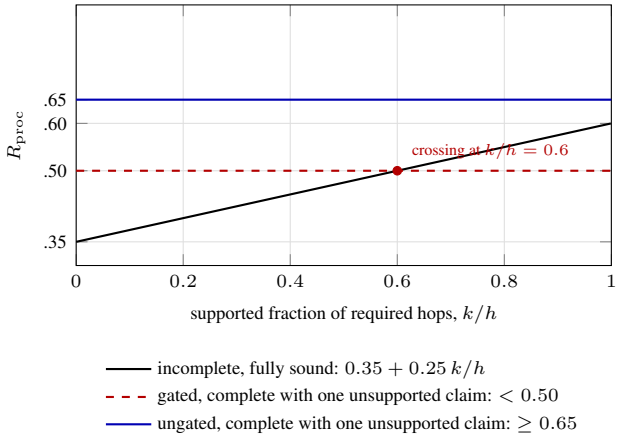


Figure 6: Reward bounds in Proposition 1. Gating can rank an establishing proof with an unsupported claim below a sound partial proof. Removing the gate satisfies P3.

B VERIFIER SPECIFICATION AND VALIDATION

The verifier follows the project's checker specification. A claim is supported when it is entailed by the full graph and schema under OWL 2 RL, rather than only the evidence shown to the model. The chain is the longest sequence of supported claims that starts at a question entity and links each object to the next subject. The contract specifies the required relation chain, its hop count and, where the graph declares types, the answer type. It is built from the question's reference path, one step per edge, and each step admits the reference relation together with the sub- and super-properties, inverses and transitive runs declared in the ontology. None of the four benchmark graphs asserts classes, so the type condition is vacuous there. 2Wiki comparison questions use a comparison contract that maps the question's wording to an operator over two branches and the dates or values they establish. The answer is the last case-insensitive "answer:" line in the trace or, failing that, its last non-empty line, and a trace is parsed when it contains at least one claim line. Citation syntax is lenient, but citation content is checked.

Implementation. The graph and ontology are held in an RDF store built on rdflib. A version counter and provenance ledger assign every asserted triple a stable identifier and record when it was added or removed. A cited identifier can therefore be resolved and classified as present, stale or unknown for the version being checked. The Rust reasoner *reasonable* materialises the OWL 2 RL closure. The version used here omits the property-chain rule, which we implement in an auxiliary layer. Both run to a fixed point so that their entailments can support each other. Materialisation is cached and invalidated after every update. Asserted and entailed triples are stored separately to distinguish directly supported steps from derived steps. Axiomatic output, such as typing every class as a class, is removed to retain only content entailments.

A proof consists of claim lines followed by an answer line. Each claim line contains a claim identifier, one or more cited identifiers of facts or earlier claims, a subject, relation and object, and for derived claims a rule name. The verifier resolves each citation against the whole graph, so a true fact outside the displayed evidence is supported, checks that the cited triples are present in the current version and closure, checks that the named rule licenses the conclusion from the cited premises, and checks closure membership of the claim's own triple. A claim that depends on an unsupported claim is unsupported. The prompt asks for citations of fact identifiers or earlier claims, but retrieval claims may also cite later or self claim identifiers, and their stated triples must still pass the closure check. On the four benchmark graphs no schema declares transitivity or property chains, so rule steps there reduce to retrieval, symmetry and inverse. It also tests for contradictions with the closure and other claims. A supported claim passes every check, a contradicted claim fails the contradic-

tion check, and other claims are unsupported. An identifier that resolves to no fact in the graph is fabricated. The verifier builds the longest supported chain from the question's entity and compares its relations with the contract to determine establishment and progress. Masking closure edges reduces coverage. Strict verification marks steps on masked edges as unsupported. Abstaining verification removes them from the soundness denominator and treats them as supported for the chain and progress checks. Hiding inverse declarations (App. C) works differently. A claim that states the hidden reverse triple is unsupported in both modes, and a forward fact used backwards in the chain remains a supported claim. Abstention changes only whether either may realise a contract step. The report fields listed in §3.1 are the only inputs to verifier-based rewards.

Checker validation. The verifier accepts all 1,552, 2,829, 3,765 and 778 reference proofs on WebQSP, CWQ, Dynamic-KGQA and 2Wiki, respectively (Table 10). An independent implementation follows the written specification, uses the pure-Python owlrl reasoner for entailment, and is cross-checked with HermiT on 560 traces requiring an entailment decision. It agrees with stored fully valid verdicts on 97.8 to 99.8% of a balanced sample of 500 accepted and rejected traces per benchmark ($\kappa = .956$ to .996). Validity verdicts differ on 15 of 2,000 traces. Including all report fields and answer correctness gives 29 disagreements. Each is explained by an alternative interpretation of the specification, with no entailment disagreement (Tables 11 and 12).

Table 10: Reference-proof acceptance and negative controls with wrong answers or swapped fact identifiers. Acceptance requires a correct answer and fully valid proof.

benchmark	eval set	n tasks	with gold proof	accepted	wrong answer accepted	swapped fact id accepted
WebQSP	eval test	1,552	1,552	1,552 (1.000)	0/1,552	0/1,552
CWQ	eval test	2,829	2,829	2,829 (1.000)	0/2,829	0/2,829
Dynamic-KGQA	eval test	3,765	3,765	3,765 (1.000)	0/3,765	0/3,765
2Wiki	eval heldout	778	778	778 (1.000)	0/778	0/778

Table 11: Agreement with an independent owlrl-based checker on 250 accepted and 250 rejected final-policy traces per benchmark, with Cohen's kappa.

benchmark	n	stored accepted	fully valid agreement	kappa	cav agreement	kappa
WebQSP	500	250	.978	.956	.980	.960
CWQ	500	250	.998	.996	.998	.996
Dynamic-KGQA	500	250	.996	.992	.996	.992
2Wiki	500	250	.998	.996	.998	.996

Table 12: The disagreements between the stored verifier and the re-check, by class. Each class is one alternate reading of the specification. None is a closure difference.

class (spec ambiguity)	WebQSP	CWQ	Dynamic-KGQA	2Wiki	all
answer-line form (answer: / final answer:) [S1]	11	0	0	11	22
repeated claim id [S3]	2	0	0	0	2
claim citing claim ids with no rule [S2]	0	0	1	1	2
malformed claim line (wrong field count) [S7]	1	1	1	0	3
total	14	1	2	12	29

C Data and experimental protocol

Graphs and contracts. The nine graphs are the four benchmark evidence collections, UMLS, CoDEx-S, the toybig graph shared with its four-hop control, a fresh graph of the same size, and the small graph shared by the toy and fact tasks. They are described below by source, size, contract derivation, and training and evaluation counts. The four benchmarks provide a separate evidence subgraph per question. From the RoG and Explore-on-Graph releases, they contain the following

training and evaluation questions. ComplexWebQuestions has 4,663 and 2,829 of 3,531, with the remainder having no path of at most three hops from a question entity to the answer in the supplied subgraph. Dynamic-KGQA has 3,783 and 3,765, WebQSP 2,708 and 1,552, and 2WikiMultihopQA 2,679, of which 1,901 form the RL pool, and 778.

The real graphs with generated questions are UMLS and CoDEx-S. UMLS contains 6,529 triples, 135 entities and 46 relations, with 168 training questions and 70 three-hop and 24 two-hop held-out questions. CoDEx-S contains 36,543 triples, 2,034 entities and 42 relations, with 340 training questions and 200 three-hop and 60 two-hop held-out questions.

One program generates the synthetic graphs. The toybig family contains 6,362 triples, 2,229 entities and 17 relations, with 2,800 training questions and 200 three-hop and 200 four-hop held-out questions. A fresh graph has the same size and schema, disjoint names and 1,600 questions per hop. It separates reasoning over evidence from recall of memorised chains. Toybig4 provides a four-hop control on the same graph. The toy and fact tasks share one small graph of 154 triples, 78 entities and 17 relations. Label-budget variants change only the warm-up set, keeping the graph and RL set fixed.

Under our shortest-path derivation, the shares of one-hop, two-hop and three-or-more-hop questions are CWQ 47.2 / 52.7 / 0.1%, Dynamic-KGQA 46.0 / 39.6 / 14.4%, WebQSP 68.3 / 31.7 / 0%, and 2Wiki 17.0 / 82.6 / 0.4%. On 2Wiki, depth counts hops per compared branch, and its depth-one questions are comparisons that need one claim per branch.

Hyperparameters. The four batch-scaled benchmark blocks use the same hyperparameters. Training uses GRPO with token-level PPO-clip loss, $\beta = 0$ and $\lambda = 1$. Each step samples 8 generations for 4 prompts, giving 32 completions. Advantages are $(r_i - \bar{r}_{\text{group}})/(\sigma + 10^{-4})$, with σ computed over the batch in the primary setting and over the group in the ablation. Advantages are not clipped, and unscorable completions receive zero advantage. We use AdamW at learning rate 10^{-5}, 5% linear warm-up followed by linear decay to zero, no weight decay, gradient norm clipping at 1.0, bf16 and full fine-tuning. vLLM generates rollouts at temperature 0.9 and top-p 1.0, with 512 new tokens for both rollout and evaluation. Evidence is capped at 60 triples, and Dynamic-KGQA uses the full released subgraph, up to 97 triples. Training runs for 3,495 steps on CWQ, 2,838 on Dynamic-KGQA, 2,031 on WebQSP and 1,425 on 2Wiki, fixed budgets corresponding to about three epochs each. CWQ and Dynamic-KGQA use 6 warm-up epochs on 2,679 2WikiMultihopQA reference proofs, with learning rate 2×10^{-5}, a cosine schedule, 10% warm-up, batch size 16, maximum length 4,096 and 1,008 steps. Evaluation uses one greedy completion per question through a vLLM path validated against the HF path. Blocks differ only in data size, training steps, evaluation set, evidence rule and warm-up, and seeds differ only in random seed. These settings use TRL 1.8.0 defaults, and warm-up copies differ by node.

Outcome with format rewards the answer only when at least one claim line parses. GATED uses $R_{\text{out}} + \lambda\, \mathbf{1}[\hat{y} = y]\, R_{\text{proc}}$ with our verifier's process term, matching the shape of K2V's Equation 8. Reference-path adds reference-path triple recall to the answer reward. EOG-RELEASE uses the released reward with zero answer weight. Step-sum adds the mean per-claim supported verdict to the answer reward. K2V-JUDGE uses K2V's Equation 8, Qwen2.5-7B-Instruct at temperature zero, and the prompt from K2V's Table 15. For correct answers, it makes one yes-or-no call per checklist criterion and returns the fraction passed. The judge sees the question, reference answer, one criterion and trace. We supply general knowledge-graph criteria because K2V's criteria concern agriculture, medicine and law. VPRM's medical risk-of-bias step-label lookup has no released code or data, so it cannot be implemented faithfully.

The reverse-relation split. The coverage experiment in §6.1 hides the declarations that allow memberOf and parentTaxon to be read in reverse. We select relations by how often training questions require their reversal, until hidden reversals affect at least 30% of the pool. Declarations for the other 40 relations remain visible. Table 13 reports the counts. About a third of questions require a hidden reversal, a third require only visible reversals, and a third require none. A claim stating a hidden reverse triple is unsupported and counts against soundness in both modes, while a forward fact used backwards keeps its support. Abstention changes only establishment, so this term is the sole difference between the training rewards. Prompts match the unsplit run, and all final policies are evaluated with the full verifier on the unsplit graph.

Table 13: Dynamic-KGQA training questions grouped by required reversals, with establishment rates for 300 sampled reference proofs under full, strict and abstaining verification. All policies are evaluated with the full verifier.

				reference proofs establish		
reference proof needs	training questions	share	sampled	full	strict	abstaining
a hidden reversal	1,343	.355	110	1.000	.000	1.000
over memberOf	1,029					
over parentTaxon	314					
only reversals the verifier can read	1,231	.325	98	1.000	1.000	1.000
no reversal	1,209	.320	92	1.000	1.000	1.000
all	3,783	1.000	300			

Tables mark planned but unscored cells as pending and unplanned cells with a dash. All runs use single nodes with 4090, A100 or Quadro GPUs. GPU hours by block will be released with the code.

D ADDITIONAL QUANTITATIVE RESULTS

This appendix reports the proof-quality metrics cited in the main text (Table 15), final rates under batch and group scaling (Tables 16 and 17), per-policy training signals (Table 18) and pass rates at one to eight samples (Table 19). Reward order is consistent across blocks. Pending cells are planned but unscored, and dashes indicate unplanned runs.

Failure modes. Table 14 reports single-claim share as a collapse marker and the share of questions answered correctly with invalid proofs as a guessing marker. The locally valid rate measures traces whose claims are all supported, distinguishing these failures from sound chains that stop short of the answer. Outcome rewards primarily fail through guessing. With warm-up, .50 of CWQ and .52 of Dynamic-KGQA questions receive correct answers with invalid proofs, compared with .15 and .07 under CDU. Without warm-up, OUTCOME-ONLY answers .94 of WebQSP questions and writes no claim lines, and answers .81 on 2Wiki with almost no locally valid traces. Verifier-based rewards can collapse without warm-up. One CDU seed and two COMBINED seeds on WebQSP, and every GATED and COU seed on 2Wiki, converge to one-claim traces. When our rewards do not collapse, about nine in ten traces are locally valid even where the correct-and-valid rate is low. Most remaining failures therefore contain only supported claims yet do not establish the correct answer. Padding is another failure mode. CDU produces 5.1 to 7.5 unused supported claims per valid proof, compared with 0.3 to 1.3 for COU (Table 7), consistent with the trade-off in §3.2. App. D, Table 15, reports support, fabrication and length by reward and benchmark. K2V-JUDGE has high seed variance, which limits interpretation of its mean. EoG-release scores zero correct-and-valid because it assigns no answer weight and generates evidence copies without answer lines. COU satisfies all five ordering properties (§3.2, App. A) but achieves lower rates than both sums on every benchmark and collapses most frequently.

Table 14: Failure markers. 1-cl.: single-claim share, dec.: fraction of questions answered correctly with invalid proofs, LV: fully supported trace share. Ranking follows Table 2, excluding answer-free rewards and three-way best ties. The single-claim share counts distinct claim labels, and in the 2Wiki K2V-judge and WebQSP outcome-with-format cells part of it is one label reused across several lines rather than a one-step trace.

	CWQ (n=2829)			Dynamic-KGQA (n=3765)			WebQSP (n=1552)			2Wiki (n=778)		
Reward	1-cl.↓	dec.↓	LV↑	1-cl.↓	dec.↓	LV↑	1-cl.↓	dec.↓	LV↑	1-cl.↓	dec.↓	LV↑
GATED	.000 ±.000	**.151*** ±.022	.968 ±.004	.000 ±.000	.073 ±.017	.972 ±.004	**.020*** ±.035	**.306*** ±.043	.953 ±.008	.999 ±.001	.775 ±.024	**.995** ±.005
K2V JUDGE	.247 ±.356	.625 ±.115	.507 ±.394	.000 ±.000	.385 ±.183	.717 ±.264	.329 ±.570	.929 ±.012	.000 ±.000	.342 ±.570	.828 ±.028	.001 ±.001
EOG PATH	.000 ±.000	.379 ±.054	.617 ±.051	.000 ±.000	.190 ±.022	.814 ±.022	**.000*** ±.000	.931 ±.004	.000 ±.000	**.000*** ±.000	.796 ±.030	.000 ±.000
EOG RELEASE	.000$^{\dagger 1}$	.000$^{\dagger 1}$	.541$^{\dagger 1}$	.000$^{\dagger 1}$	.000$^{\dagger 1}$	.743$^{\dagger 1}$	.000$^{\dagger 1}$	.001$^{\dagger 1}$	.000$^{\dagger 1}$	.000$^{\dagger 1}$	.000$^{\dagger 1}$	.000$^{\dagger 1}$
STEP SUM	.034 ±.057	.465 ±.034	**.990*** ±.002	.057 ±.067	.412 ±.041	**.991** ±.001	.503 ±.500	.888 ±.018	**.998** ±.001	.298 ±.516	.740 ±.042	.982 ±.012
OUTCOME	.007 ±.011	.504 ±.051	.849 ±.044	.000 ±.001	.521 ±.350	.614 ±.532	**.000*** ±.000	.943 ±.004	.000 ±.000	**.233*** ±.401	.810 ±.005	.000 ±.001
OUTCOME + FMT	.002 ±.003	.481 ±.010	.867 ±.076	.004 ±.007	.341 ±.041	.965 ±.011	.654 ±.546	.915 ±.025	.399 ±.340	.319 ±.344	.794 ±.024	.104 ±.169
CDU (OURS)	.000 ±.000	**.152*** ±.012	.956 ±.013	.000 ±.000	**.070*** ±.009	.969 ±.002	.333 ±.577	**.325*** ±.116	.930 ±.042	**.000*** ±.001	**.480*** ±.015	.892 ±.037
COMBINED (OURS)	.000 ±.000	.169 ±.035	.976 ±.006	.000 ±.000	**.067*** ±.015	.978 ±.003	.666 ±.577	.365 ±.105	.976 ±.013	.667 ±.575	**.638*** ±.243	.974 ±.038
COU	.206 ±.048	.244 ±.013	**.984*** ±.001	.002 ±.003	.175 ±.016	.978 ±.002	.807 ±.332	.368 ±.072	.959 ±.022	1.000 ±.001	.791 ±.018	.952 ±.007

Model size and repair. The trained 1.5B model outperforms the untrained 7B model in correct-and-valid rate, and verifier feedback improves models of at least 3B. Untrained Qwen2.5 models from 0.5B to 7B reach .006 to .30 correct-and-valid at .16 to .64 answer accuracy. Gemini 3.6 Flash and Claude Sonnet 5 reach .72 to .94 zero-shot. With warm-up, the trained 1.5B model reaches .70 to .86, above the untrained 7B model and close to the frontier models (Fig. 3, Table 3). Two verifier-feedback turns on rejected traces improve every model of 3B or larger. The 7B model gains .10 to .13 and Sonnet .07 to .13. Trained policies improve modestly, with no decrease in shipped-answer accuracy (Table 5). The shipped answer is the last validated turn, or the first if none validates. This experiment measures the benefit of our verifier's feedback within an existing repair loop.

Table 15: Proof quality of greedy batch-scaled policies (mean ± seed SD). FV: fully valid rate; LV: locally valid rate; VS: valid-step rate; UT: unsupported-transition rate; Fab: fabrication rate; Len: claims per trace; Acc: answer accuracy; Pad: unused supported claims per fully valid trace. Superscripts give seed counts; — marks unmeasured cells; n/a marks padding undefined because no trace is fully valid.

Reward	FV	LV	VS	UT	Fab	Len	Acc	Pad
CWQ (cwqB4, n=2829)								
CDU	.755±.007	.956±.013	.994±.001	.003±.001	.010±.002	9.06±2.78	.856±.014	7.47±2.83
COMBINED	.738±.033	.976±.006	.995±.000	.002±.001	.007±.000	5.78±1.70	.855±.003	4.08±1.65
COU	.653±.019	.984±.001	.992±.001	.004±.001	.006±.002	1.96±0.11	.847±.010	0.37±0.08
GATED	.760±.021	.968±.004	.995±.000	.002±.000	.009±.002	6.68±1.14	.859±.006	4.92±1.00
K2V-STYLE JUDGE	.256±.128	.507±.394	.764±.183	.108±.090	.062±.044	3.01±1.48	.865±.003	0.99±0.78
EOG PATH	.520±.055	.617±.051	.950±.014	.022±.005	.151±.031	10.81±0.85	.864±.005	9.20±0.85
EOG RELEASE	.000$^{\dagger 1}$	.541$^{\dagger 1}$	.958$^{\dagger 1}$	.026$^{\dagger 1}$	.210$^{\dagger 1}$	16.14$^{\dagger 1}$	.000$^{\dagger 1}$	n/a
VPRM STEP SUM	.426±.033	.990±.002	.995±.001	.002±.000	.003±.000	2.08±0.20	.868±.007	0.58±0.20
OUTCOME	.393±.058	.849±.044	.934±.025	.018±.002	.038±.008	2.52±0.58	.879±.006	0.87±0.45
OUTCOME+FMT	.421±.007	.867±.076	.945±.017	.010±.002	.017±.004	2.43±0.68	.882±.004	0.81±0.44
Dynamic-KGQA (dqB3, n=3765)								
CDU	.880±.010	.969±.002	.996±.000	.002±.000	.009±.002	8.47±0.11	.926±.005	6.76±0.10
COMBINED	.877±.012	.978±.003	.997±.001	.002±.001	.008±.003	7.89±2.32	.918±.005	6.18±2.31
COU	.772±.017	.978±.002	.993±.002	.003±.001	.007±.001	2.94±0.12	.922±.008	1.31±0.10
GATED	.869±.018	.972±.004	.996±.000	.002±.000	.011±.001	7.90±1.89	.917±.009	6.13±1.88
K2V-STYLE JUDGE	.543±.197	.717±.264	.911±.092	.065±.093	.040±.018	3.94±0.33	.913±.011	1.80±0.24
EOG PATH	.740±.018	.814±.022	.979±.004	.013±.002	.090±.024	11.38±0.62	.911±.007	9.68±0.64
EOG RELEASE	.000$^{\dagger 1}$	.743$^{\dagger 1}$	.980$^{\dagger 1}$	.017$^{\dagger 1}$	.109$^{\dagger 1}$	16.28$^{\dagger 1}$	.000$^{\dagger 1}$	n/a
VPRM STEP SUM	.524±.045	.991±.001	.996±.000	.002±.001	.005±.002	2.04±0.15	.918±.003	0.77±0.10
OUTCOME	.413±.362	.614±.532	.650±.563	.340±.571	.016±.014	2.04±1.94	.922±.004	1.52±0.88$^{\dagger 2}$
OUTCOME+FMT	.595±.043	.965±.011	.987±.002	.006±.001	.015±.004	2.55±0.38	.917±.006	1.14±0.30
WebQSP (webqspB, n=1552)								
CDU	.609±.120	.930±.042	.985±.006	.005±.003	.016±.008	6.33±4.67	.922±.005	5.12±4.50
COMBINED	.568±.117	.976±.013	.987±.006	.008±.005	.010±.002	2.81±3.13	.921±.013	1.65±2.86
COU	.559±.072	.959±.022	.975±.008	.010±.005	.012±.005	1.55±0.95	.910±.005	0.30±0.52
GATED	.635±.047	.953±.008	.989±.003	.005±.002	.015±.004	4.97±2.29	.930±.004	3.55±2.12
K2V-STYLE JUDGE	.000±.000	.000±.000	.000±.001	.918±.072	.014±.007	2.33±1.53	.929±.012	n/a
EOG PATH	.000±.000	.000±.000	.054±.075	.064±.020	.224±.016	9.30±0.79	.931±.004	n/a
EOG RELEASE	.000$^{\dagger 1}$	.000$^{\dagger 1}$	.030$^{\dagger 1}$	.088$^{\dagger 1}$	.340$^{\dagger 1}$	22.17$^{\dagger 1}$	.001$^{\dagger 1}$	n/a
VPRM STEP SUM	.045±.017	.998±.001	.998±.001	.000±.000	.000±.000	2.17±1.62	.931±.002	1.05±1.68
OUTCOME	.000±.000	.000±.000	.000±.000	1.000±.000	.000±.000	0.00±0.00	.943±.004	n/a
OUTCOME+FMT	.026±.026	.399±.340	.492±.198	.037±.025	.025±.032	1.92±1.54	.940±.004	0.21±0.36
2Wiki (eogB3, n=778)								
CDU	.294±.012	.892±.037	.982±.007	.005±.002	.025±.008	7.49±3.67	.758±.014	5.18±3.60
COMBINED	.137±.237	.974±.038	.994±.004	.002±.002	.006±.009	3.67±4.62	.771±.024	6.68$^{\dagger 1}$
COU	.000±.000	.952±.007	.952±.007	.003±.003	.003±.003	1.00±0.00	.791±.018	n/a
GATED	.000±.000	.995±.005	.995±.005	.001±.002	.001±.001	1.00±0.00	.775±.024	n/a
K2V-STYLE JUDGE	.000±.001	.001±.001	.014±.025	.627±.447	.070±.025	1.79±0.71	.829±.027	0.00$^{\dagger 1}$
EOG PATH	.000±.000	.000±.000	.037±.020	.165±.222	.087±.055	12.01±0.68	.796±.030	n/a
EOG RELEASE	.000$^{\dagger 1}$	.000$^{\dagger 1}$	.119$^{\dagger 1}$	.046$^{\dagger 1}$	.204$^{\dagger 1}$	19.52$^{\dagger 1}$	.000$^{\dagger 1}$	—
VPRM STEP SUM	.029±.028	.982±.012	.995±.002	.000±.000	.003±.003	4.84±3.89	.768±.016	2.80±3.52
OUTCOME	.000±.000	.000±.001	.002±.003	.897±.177	.317±.548	0.91±1.04	.810±.005	n/a
OUTCOME+FMT	.001±.001	.104±.169	.190±.191	.223±.175	.123±.135	2.33±1.20	.795±.025	0.00±0.00$^{\dagger 2}$

Batch and group advantage scaling. The main comparison uses batch scaling, and Table 8 compares both scalings on WebQSP. Tables 16 and 17 give final correct-and-valid rates for each scaling. Table 18 gives the per-policy training signals for WebQSP and 2Wiki summarised in Table 8.

Advantage scaling. Group-scaled runs lose reward diversity under every reward (Table 8). It divides each completion's centred reward by the standard deviation of its group of eight (App. C). Components shared by all eight completions cancel, so their magnitude has no effect. Groups with identical rewards, called dead groups, carry no gradient. Groups with nearly identical rewards divide by a near-zero deviation, amplifying small differences. In one synthetic run, groups whose eight rewards differed by less than 0.01 supplied .27 of the gradient magnitude under group scaling and

.004 under batch scaling. Every WebQSP reward loses diversity under group scaling. Over the last third of training, dead-group share reaches .576 for CDU, .659 for GATED, .877 for COU and .968 to .973 for answer-based rewards. Under batch scaling, CDU, COMBINED, COU and GATED remain at or below .07, because division by the batch standard deviation keeps near-ties small. The severity depends on the reward. COU collapses on every seed, with single-claim share .989 and entropy .009. CDU and GATED retain entropy near .03 without collapse. Under batch scaling, groups retain variance but collapse varies by seed. One CDU seed, two COMBINED seeds and two COU seeds collapse, and no GATED seed does. GATED leads CDU under batch scaling (.624 against .597), while CDU leads under group scaling (.661 against .588). On Dynamic-KGQA, the sums and GATED change by about .01 between scalings, while COU and outcome move by .16 and .23. CWQ favours batch scaling for every reward except OUTCOME WITH FORMAT (Appendix D, Tables 16 and 17). Table 18 gives the per-seed training signals.

Table 16: Correct-and-valid rates under batch scaling (mean $\pm$ seed SD). Superscripts give seed counts below three, and dashes mark unplanned runs.

Reward	WebQSP (n=1552)	Dynamic-KGQA (n=3765)	CWQ (n=2829)
OURS: DENSE (CDU)	.597 $\pm$ .120	.856 $\pm$ .012	.704 $\pm$ .010
OURS: COMBINED	.556 $\pm$ .116	.851 $\pm$ .012	.685 $\pm$ .032
OURS: ORDERED	.542 $\pm$ .076	.747 $\pm$ .020	.603 $\pm$ .022
K2V-SHAPE GATE	.624 $\pm$ .047	.844 $\pm$ .017	.708 $\pm$ .024
EOG PATH REWARD	.000 $\pm$ .000	.721 $\pm$ .019	.485 $\pm$ .052
EOG RELEASE REWARD	.000$^{\dagger 1}$	.000$^{\dagger 1}$	.000$^{\dagger 1}$
VPRM STEP SUM	.044 $\pm$ .016	.506 $\pm$ .044	.403 $\pm$ .028
OUTCOME ONLY	.000 $\pm$ .000	.401 $\pm$ .352	.374 $\pm$ .056
OUTCOME + FORMAT	.026 $\pm$ .026	.576 $\pm$ .045	.401 $\pm$ .009

Table 17: Correct-and-valid rate of the greedy final policy under group advantage scaling, on the WebQSP, Dynamic-KGQA and CWQ group blocks.

Reward	WebQSP (n=1552)	Dynamic-KGQA (n=3765)	CWQ (n=2829)
OURS: DENSE (CDU)	.661 $\pm$ .010	.867 $\pm$ .005	.669 $\pm$ .010
OURS: COMBINED	.636 $\pm$ .051	.847 $\pm$ .006	.657 $\pm$ .023
OURS: ORDERED	.475 $\pm$ .013	.589 $\pm$ .030	.462 $\pm$ .023
K2V-SHAPE GATE	.588 $\pm$ .113	.841 $\pm$ .013	.655 $\pm$ .011
EOG PATH REWARD	.000 $\pm$ .000	.754 $\pm$ .023	.387 $\pm$ .094
EOG RELEASE REWARD	–	.000$^{\dagger 1}$	.000 $\pm$ .000$^{\dagger 2}$
VPRM STEP SUM	.041 $\pm$ .008	.553 $\pm$ .081	.292 $\pm$ .096
OUTCOME ONLY	.014 $\pm$ .025	.635 $\pm$ .027	.272 $\pm$ .236
OUTCOME + FORMAT	.032 $\pm$ .014	.655 $\pm$ .057	.432 $\pm$ .031

Table 8 averages training signals over available logs, with up to three seeds under batch scaling and two under group scaling on WebQSP, one seed on 2Wiki, and per-arm counts in Table 18. Table 18 reports individual policies. Correct-and-valid rate and single-claim share use greedy final traces (WebQSP $n = 1{,}552$, 2Wiki $n = 778$). Single-claim share includes traces with at most one claim. Dead-group share counts prompt groups whose eight rewards tie. Entropy is mean token entropy, and length is mean completion length in tokens. Early, middle and late values average the corresponding thirds of training. One-claim onset is the step after which rolling completion length remains below 60 tokens, and a dash indicates no onset. Policies without logs are omitted.

Table 18: Per-seed training signals for WebQSP and 2Wiki under batch and group scaling. Column definitions and averaging windows are given in the preceding text.

				dead groups			entropy		length		
reward	seed	cav.	1-cl.	early	mid	late	early	late	early	late	one-claim from
WebQSP, batch scaling											
GATED REF.	0	.570	.061	.018	.028	.061	.064	.028	60	87	–
GATED REF.	1	.644	.000	.012	.033	.092	.077	.026	106	154	–
GATED REF.	2	.658	.000	.013	.042	.054	.089	.031	113	213	–
EOG PATH	0	.000	.000	.022	.130	.140	.133	.094	232	285	–
EOG PATH	1	.000	.000	.016	.121	.182	.184	.119	193	253	–
EOG PATH	2	.000	.000	.036	.123	.147	.194	.119	250	314	–
EOG RELEASE	0	.000	.000	.059	.330	.409	.167	.096	490	508	–
STEP SUM	0	.061	.000	.181	.557	.648	.082	.037	169	136	–
OUTCOME	0	.000	1.000	.481	.720	.798	.071	.023	26	6	85
OUTCOME + FMT	0	.055	1.000	.373	.673	.749	.134	.081	67	37	420
OUTCOME + FMT	1	.013	.938	.320	.652	.727	.183	.144	101	82	–
OUTCOME + FMT	2	.008	.026	.348	.579	.679	.178	.173	137	118	–
CDU (OURS)	0	.631	.000	.002	.007	.028	.084	.034	224	308	–
CDU (OURS)	1	.463	1.000	.000	.025	.047	.083	.007	144	35	590
CDU (OURS)	2	.697	.000	.000	.024	.025	.085	.026	209	261	–
COMBINED (OURS)	0	.490	1.000	.012	.077	.067	.059	.008	54	35	100
COMBINED (OURS)	1	.488	1.000	.009	.056	.067	.053	.008	52	35	85
COMBINED (OURS)	2	.689	.000	.025	.042	.035	.059	.047	57	214	–
DENSE (GATED)	0	.655	.000	.003	.024	.038	.079	.034	182	249	–
COU	0	.499	.999	.004	.047	.057	.056	.008	55	35	105
COU	1	.497	1.000	.012	.043	.064	.056	.005	52	35	110
COU	2	.630	.424	.006	.009	.009	.079	.029	85	91	–
ORDERED (GATED)	0	.617	.504	.000	.013	.039	.074	.016	81	61	–
ORDERED ANY-HOP UNGATED	0	.629	.438	.002	.006	.018	.076	.028	101	104	–
ORDERED ANY-HOP	0	.574	.561	.003	.010	.029	.064	.018	59	54	2030
WebQSP, group scaling											
GATED REF.	0	.608	.000	.574	.642	.685	.066	.029	70	200	–
GATED REF.	1	.689	.002	.590	.654	.632	.061	.033	67	158	–
EOG PATH	0	.000	.000	.374	.570	.629	.162	.090	235	295	–
EOG PATH	1	.000	.000	.430	.660	.732	.171	.149	203	280	–
STEP SUM	0	.049	1.000	.729	.954	.973	.056	.015	66	37	135
STEP SUM	1	.039	.999	.723	.956	.983	.061	.012	69	38	145
OUTCOME	0	.000	.048	.767	.933	.949	.149	.135	146	137	–
OUTCOME	1	.000	1.000	.852	.975	.987	.067	.020	24	6	75
OUTCOME + FMT	0	.022	.001	.769	.937	.961	.168	.130	132	145	–
OUTCOME + FMT	1	.026	.999	.804	.966	.985	.102	.033	60	37	120
CDU (OURS)	0	.672	.000	.262	.490	.542	.082	.024	187	313	–
CDU (OURS)	1	.658	.000	.303	.530	.610	.075	.024	169	264	–
COMBINED (OURS)	0	.664	.000	.348	.563	.670	.079	.029	192	202	–
COMBINED (OURS)	1	.577	.236	.577	.626	.791	.059	.020	56	77	–
DENSE (GATED)	0	.671	.000	.299	.513	.587	.081	.029	173	285	–
COU	0	.490	1.000	.516	.829	.891	.059	.008	54	35	110
COU	1	.465	.977	.413	.811	.863	.068	.009	59	36	205
ORDERED (GATED)	0	.495	1.000	.488	.842	.903	.066	.006	51	35	85
ORDERED (GATED)	1	.493	1.000	.487	.831	.897	.053	.005	51	35	100
ORDERED ANY-HOP UNGATED	0	.478	1.000	.471	.810	.882	.059	.007	52	35	110
ORDERED ANY-HOP UNGATED	1	.454	1.000	.506	.827	.888	.056	.006	50	35	80
ORDERED ANY-HOP	0	.475	1.000	.458	.822	.868	.056	.007	51	35	85
2Wiki, batch scaling											
GATED REF.	0	.000	1.000	.085	.173	.325	.090	.052	53	36	390
EOG PATH	0	.000	.000	.000	.006	.006	.178	.126	223	312	–
EOG RELEASE	0	.000	.001	.004	.118	.138	.192	.109	489	510	–
STEP SUM	0	.027	.000	.023	.128	.208	.091	.034	129	143	–
OUTCOME	0	.000	.003	.074	.190	.325	.240	.147	101	78	–
OUTCOME + FMT	0	.001	.684	.094	.232	.365	.174	.134	110	39	315
CDU (OURS)	0	.279	.000	.000	.000	.000	.126	.057	148	210	–
COMBINED (OURS)	0	.397	.003	.006	.002	.002	.094	.037	165	275	–
COU	0	.000	.999	.002	.011	.013	.104	.014	71	36	125
ORDERED (GATED)	0	.000	.999	.002	.036	.050	.101	.018	74	35	145
ORDERED ANY-HOP UNGATED	0	.000	1.000	.004	.042	.052	.105	.038	81	51	705
2Wiki, group scaling											
GATED REF.	0	.030	.738	.612	.772	.838	.084	.027	81	45	140
OUTCOME	0	.000	1.000	.773	.917	.952	.068	.007	52	7	100
OUTCOME + FMT	0	.000	.105	.723	.819	.869	.144	.088	132	136	–
COMBINED (OURS)	0	.000	.999	.528	.814	.894	.100	.040	84	36	205
DENSE (GATED)	0	.332	.000	.081	.245	.271	.108	.038	280	347	–
COU	0	.000	1.000	.435	.765	.818	.073	.013	66	37	95
ORDERED (GATED)	0	.000	1.000	.429	.761	.853	.071	.006	67	37	105
ORDERED ANY-HOP UNGATED	0	.000	1.000	.524	.817	.884	.059	.005	64	37	80
ORDERED ANY-HOP	0	.000	1.000	.442	.732	.780	.067	.012	66	37	100

Sampling. Table 19 gives CoT-Pass rates at one, two, four and eight samples for the ten rewards in the main table. Sampling uses temperature 0.9. K2V-JUDGE runs use the same data, and results are means over the same seeds as the greedy tables.

Table 19: CoT-Pass is the fraction of questions with a correct and valid trace among k samples at temperature 0.9. Answer-only pass@8 is also shown. Means use the same seed IDs as the greedy results; decoding differs. Pending and dashes follow App. C.

Benchmark	Reward	seeds	CoT-P@1	CoT-P@2	CoT-P@4	CoT-P@8	gain	pass@8
CWQ (n=2829)								
	OURS: DENSE (CDU)	3	.693	.750	.787	.814	+.121	.904
	OURS: COMBINED	3	.678	.727	.762	.791	+.113	.903
	OURS: ORDERED	3	.598	.623	.643	.660	+.062	.878
	K2V-SHAPE GATE	3	.698	.751	.788	.817	+.119	.909
	K2V-STYLE LLM JUDGE	3	.236	.272	.307	.341	+.105	.903
	OUTCOME ONLY	3	.355	.413	.461	.503	+.148	.915
	OUTCOME + FORMAT	3	.376	.437	.488	.533	+.157	.913
	EOG PATH REWARD	3	.436	.577	.685	.759	+.323	.912
	EOG RELEASE REWARD	1	.000	.000	.001	.001	+.001	.001
	VPRM STEP SUM	3	.400	.433	.461	.487	+.087	.897
Dynamic-KGQA (n=3765)								
	OURS: DENSE (CDU)	3	.846	.888	.912	.928	+.082	.956
	OURS: COMBINED	3	.844	.880	.902	.919	+.074	.952
	OURS: ORDERED	3	.743	.765	.781	.793	+.049	.946
	K2V-SHAPE GATE	3	.833	.875	.900	.917	+.084	.952
	K2V-STYLE LLM JUDGE	3	.527	.591	.646	.692	+.165	.956
	OUTCOME ONLY	3	.392	.426	.452	.473	+.082	.947
	OUTCOME + FORMAT	3	.571	.606	.633	.656	+.085	.946
	EOG PATH REWARD	3	.666	.789	.860	.900	+.233	.957
	EOG RELEASE REWARD	1	.000	.000	.001	.001	+.001	.002
	VPRM STEP SUM	3	.505	.524	.540	.554	+.049	.943
WebQSP (n=1552)								
	OURS: DENSE (CDU)	3	.580	.647	.694	.729	+.149	.950
	OURS: COMBINED	3	.539	.575	.600	.621	+.082	.942
	OURS: ORDERED	3	.534	.560	.580	.596	+.062	.934
	K2V-SHAPE GATE	3	.615	.672	.713	.748	+.133	.956
	K2V-STYLE LLM JUDGE	3	.000	.000	.000	.000	+.000	.963
	OUTCOME ONLY	3	.000	.000	.000	.000	+.000	.955
	OUTCOME + FORMAT	3	.022	.037	.057	.084	+.062	.963
	EOG PATH REWARD	3	.000	.000	.000	.000	+.000	.961
	EOG RELEASE REWARD	1	.000	.000	.000	.000	+.000	.048
	VPRM STEP SUM	3	.045	.061	.079	.100	+.055	.950
2Wiki (n=778)								
	OURS: DENSE (CDU)	3	.265	.340	.405	.457	+.191	.827
	OURS: COMBINED	3	.125	.153	.177	.198	+.074	.842
	OURS: ORDERED	3	.000	.000	.000	.000	+.000	.840
	K2V-SHAPE GATE	3	.000	.000	.000	.000	+.000	.833
	K2V-STYLE LLM JUDGE	3	.000	.000	.000	.000	+.000	.899
	OUTCOME ONLY	3	.000	.000	.000	.000	+.000	.867
	OUTCOME + FORMAT	3	.003	.006	.011	.018	+.015	.894
	EOG PATH REWARD	3	.000	.000	.000	.000	+.000	.905
	EOG RELEASE REWARD	1	.000	.000	.000	.000	+.000	.005
	VPRM STEP SUM	3	.022	.035	.052	.072	+.049	.836

E TRAINING DYNAMICS AND FAILURE CASES

Fig. 7 shows single-claim shares in greedy evaluation on WebQSP and Dynamic-KGQA, and in training rollouts on WebQSP and 2Wiki, for the displayed scaling configurations. Fig. 8 reports adaptive WebQSP rewards, with the same settings as the original WebQSP runs except for the reward. No reward collapses with warm-up. Without it, ordered collapses in every block, dense ungated on one WebQSP seed, one 0.5B UMLS seed and all 0.5B CoDEx-S seeds, COMBINED on two 0.5B CoDEx-S seeds, and GATED on every 2Wiki seed. Per-block training curves will be released with the code.

The one-claim collapse phenomenon. Without warm-up, some policies reduce their proofs to one claim and remain there. We classify a run as collapsed if its single-claim share stays at or above .8 over the final tenth of training, and as a dip if it reaches .8 earlier and then falls (Fig. 5). Training rollouts follow the same pattern across blocks. The initial policy produces eight to fourteen claims per trace, of which .62 to .86 are wrong, and answers .03 to .30 of questions correctly. Every one of the 18 logged runs shortens its traces during the first fifty steps, usually to 2.6 to 6.0 claims. Collapsing runs then fall to one claim at a median step of 110 and do not recover. They have collapsed by the first scored checkpoint, and training rollouts agree with greedy evaluation at the checkpoints (App. E, Fig. 7). The exception to early onset is one WebQSP CDU seed, which collapses near step 570 (step 590 by the length criterion of Table 18). Its completion length falls from 306 to 35 tokens within about 45 steps. Seven of the runs in Fig. 5 reach .8 and later fall, one of them shown to step 1,573, so a temporary reduction to one claim does not determine the final state.

Three pressures towards collapse. Three factors favour shorter traces at the start of training. The first is the penalty for wrong claims. While most claims are wrong, deleting claims increases reward more than extending the chain. Recomputing both changes from logged reward components shows a larger gain from truncation on .90 to .97 of CDU traces with wrong answers and a mixture of supported and wrong claims, and on .97 to .99 of such traces under the ordered reward. The rewards differ when no claim is supported. The ordered reward penalises the number of wrong claims per hop, so deletion down to one wrong claim increases reward on .90 to .99 of these traces. Under the sums' bounded soundness fraction, deletion helps only when it removes a fabricated or contradicted claim, and no contradiction occurs on these graphs.

The second factor is the answer term, which continues to reward a policy that produces one claim. Collapsed 2Wiki policies answer .80 to .85 of questions correctly, as do non-collapsed policies, although no 2Wiki question can be proved with one claim. One collapsed run converges to a single wrong claim. The third factor is specific to WebQSP, where 68% of questions are one-hop. A one-claim proof is complete for these questions, and the dense reward still credits the verified hop on the others. Collapse therefore has little effect on mean training reward. Two seeds of the same reward reach 1.80, one collapsed and one non-collapsed. Final training reward does not distinguish these states, whereas per-seed held-out correct-and-valid rate does.

Once a policy produces one claim, its completions become similar. On the collapsed WebQSP seed, token entropy at the final logged step is .000, compared with .024 for GATED on the same seed. Under group scaling, tied prompt groups carry no gradient (§6.1), preventing recovery. Most claims are still wrong at collapse onset: .53 to .86 in the twenty preceding steps for seven of eight collapsed runs. Two runs that later recover also have this pattern, so the final outcome depends on the updates during this period. Reward shape affects susceptibility. Counting final policies with at least .9 single-claim greedy traces over the no-warm-up blocks, including group scaling and one 0.5B 2Wiki block, ordered rewards collapse in 17 of 21 runs, COMBINED in 4 of 8, GATED in 2 of 8 and CDU in 1 of 8. No run with warm-up collapses. Warm-up removes most wrong claims, so deletion has little to gain. No ablation run with warm-up, with group scaling for the sums, or without one-hop questions collapses. Removing one-hop questions (seed 1, 657 updates) reduces the rate to .372, compared with .696 for non-collapsed seed 2 trained on all questions for 2,031 updates (§6.1).

Adaptive rewards. Adaptive rewards address the wrong-claim penalty. They multiply it by the squared supported-claim share of the last ten training steps, so deleting a claim has little effect early in training when most claims are wrong. Fabricated traces are no longer capped. Their fabricated claims are charged as unsupported claims under the same scale, plus a fixed penalty of 0.10 per trace. On WebQSP without warm-up, neither adaptive variant collapses at the final checkpoint across three seeds. ADAPTIVE DENSE reaches .666, compared with .597 for CDU and .624 for GATED, and exceeds GATED on every seed. ADAPTIVE ORDERED reaches .612, compared with .542 for COU. Gains concentrate on previously collapsed seeds. Other seeds change by $-.011$ to $+.029$, while seed ranges narrow from .234 to .037 for dense and from .133 to .011 for ordered (App. E, Fig. 8). ADAPTIVE DENSE still produces 7 to 11 claims per answer, and ADAPTIVE ORDERED's single-claim share rises to .17 to .47 without final collapse. On the group-scaled Dynamic-KGQA block with warm-up, ADAPTIVE ORDERED reaches .580 compared with .555 for COU, below the sums at .84 to .86 on one seed. On the group-scaled 2Wiki block without labels, ADAPTIVE ORDERED collapses to .000, while ADAPTIVE DENSE reaches .386 compared with .402 for CDU without collapsing, on

one seed with $n = 778$. These results are consistent with the adaptive penalty removing the wrong-claim pressure while the answer-term pressure remains. The experiment changes the fabrication handling at the same time, so further experiments are needed to isolate the two. Future work could withhold the answer reward until a proof reaches the required depth or vary the process weight during training. An entropy bonus does not address this problem. On the synthetic graph, it halves the share of groups with identical completions but leaves the share of groups with near-tied rewards (standard deviation below .01) unchanged.

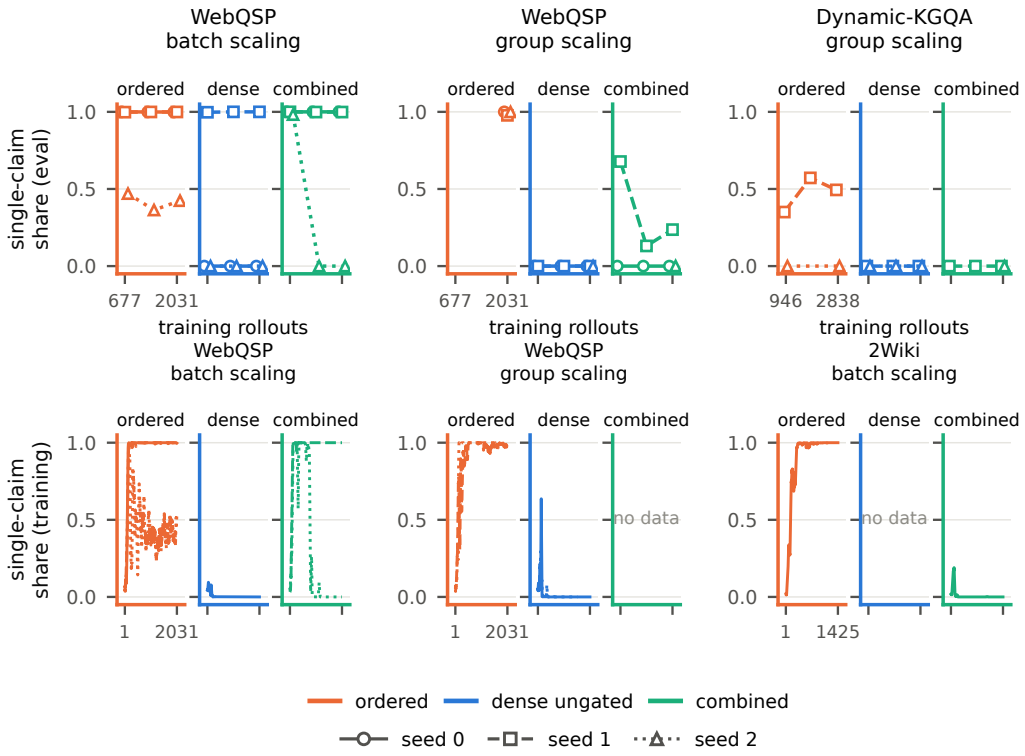


Figure 7: Single-claim shares in greedy evaluation (top) and training (bottom) for the indicated benchmarks and scalings. Training curves use 25-step means. Line styles identify seeds, and missing logs are marked “no data”.

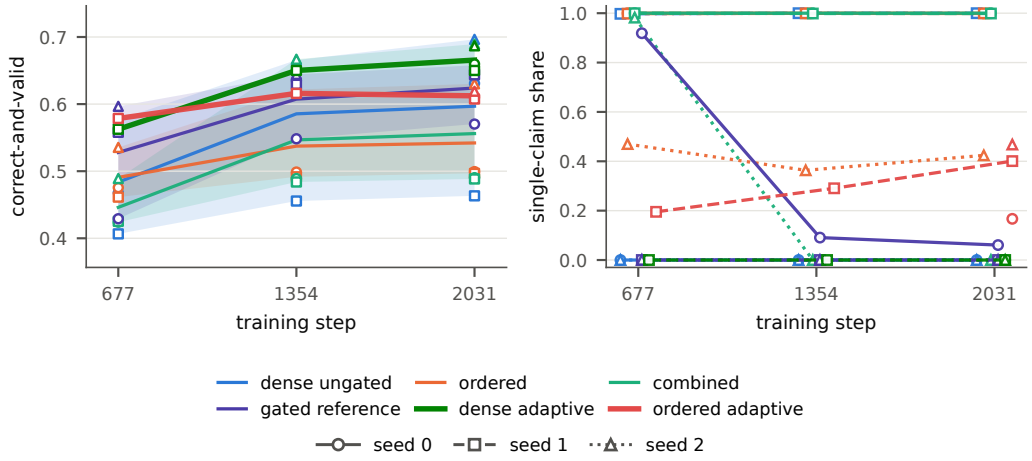


Figure 8: Adaptive rewards on WebQSP without warm-up. Left: correct-and-valid means and seed ranges. Right: single-claim shares by seed. Adaptive curves use one seed at intermediate checkpoints and three at the final checkpoint.

F SYNTHETIC AND SMALL-GRAPH EXPERIMENTS

Small and synthetic graphs. Experiments on UMLS, CoDEx-S and synthetic graphs preceded the benchmark experiments. Three findings extend to the benchmarks. Without proof labels, verifier-based rewards outperform answer-based rewards on the synthetic graphs and in the 1.5B small-graph blocks. The outcome mean is higher at the 25% synthetic label budget, and at 0.5B on CoDEx-S CDU, COU and GATED score zero while COMBINED and REFERENCE-PATH are positive. Policies learn valid proofs without reference proofs, reaching 0.38 on held-out four-hop synthetic questions (Table 20) and 0.278 on 2Wiki. Collapse without warm-up varies across seeds. At 0.5B on CoDEx-S and UMLS, collapse under our rewards depends on the seed and graph (App. F, Table 23). The coverage result differs. Abstaining on masked relations recovers most or all of the loss on the small synthetic graphs except at 0.5B on the fact graph, but on Dynamic-KGQA hiding the reversals lowers the mean from .856 to .838 and abstention at .836 does not recover it (§6.1).

The synthetic results use an earlier protocol. The verifier predates the 2026-09-12 inverse-step change, completions are capped at 160 tokens unless stated otherwise, and runs use supervised warm-up on their own graph unless stated otherwise. Evaluation sets are small. These results support comparisons within tables, but not direct comparisons of magnitudes across tables. Regenerated values replace discussion-point values that did not reproduce (148 of 156 meet the audit's precision-dependent agreement criterion, and 147 differ by at most .005). One program generates the synthetic graphs, allowing depth, coverage, label budget and shortcuts to vary independently while holding the graph fixed, except in the fresh-graph condition. The benchmarks do not support these controls. The three tables cited in the main text follow. Full grids, including 0.5B and 7B runs, process-weight variations, the fresh-graph verifier audit, and shortcut and depth studies, will be released with the code.

Learning without labels. Without reference proofs, the verifier-supervised policy reaches 0.38 correct-and-valid on held-out four-hop questions at a 160-token cap, 0.95 when rerun at 512 tokens, and 0.46 on the fresh graph. Outcome reward remains at 0.000 (Table 20). Warm-up uses the stated share of reference proofs, followed by RL on the full question pool. Zero-label runs receive no reference proofs.

Table 20: Correct-and-valid rates for 1.5B policies with varying reference-proof warm-up on the large synthetic graph. Zero-label runs receive no reference proofs. Mean ± SD over three seeds unless marked, under the protocol of App. F.

Gold proofs in warm-up	in-graph 4-hop: SFT	in-graph 4-hop: outcome	in-graph 4-hop: ours	fresh 4-hop: SFT	fresh 4-hop: outcome	fresh 4-hop: ours
0% (no warm-up)	.000±.000	.000±.000	.377±.034	.000[1]	.000±.000	.458±.032
0% (no warm-up), 512-token rerun	.000±.000	.000±.000	.950±.009	—	.000[1]	—
25% (700 proofs)	.597±.086	.657±.193	.608±.320	.613±.087[2]	.648±.201	.590±.305
100% (2,800)	.503±.203	.248±.201	.445±.221	.514±.189	.273±.175	.416±.194
200% (5,600; SFT only)	.740±.044	—	—	.754±.042	—	—
max (8,540)	.613±.231	.593±.260	.622±.203	—	—	—

Verifier coverage and its remedies. We test whether masking relations harms training and whether inverse-coverage reweighting of the process term or an abstaining verifier recovers the loss. On the small graph at 0.5B, masking a quarter of relations reduces the rate from 0.70 to 0.29. Abstaining on unseen steps restores it to 0.80, above the full-verifier rate. At 0.5B on the fact graph abstention recovers only part of the loss (.226 to .102 to .130). At 1.5B the loss is small on the toy graph and .19 on the fact graph, and abstention recovers both, while the toybig sets do not lose under masking and UMLS recovers little (Table 21). Synthetic tables report means and sample standard deviations over three seeds unless a superscript gives another count, and a dash marks a condition not run.

Table 21: Correct-and-valid rates with full coverage or a quarter of relations masked, using strict verification, abstention or inverse-coverage reweighting of the process term. Mean ± SD over three seeds unless marked, under the protocol of App. F.

Graph / set	Model	n	full	blind .75	abstain .75	reweight .75
toy 3-hop	0.5B	22	.697±.139	.288±.274	.803±.069	.545±.091
toy 3-hop	1.5B	22	1.000±.000	.970±.052	1.000±.000	.833±.114
fact held-out	0.5B	59	.226±.080	.102±.088	.130±.113	.102±.090
fact held-out	1.5B	59	.712±.029	.520±.085	.723±.060	.650±.093
toybig 4-hop	1.5B	200	.445±.221	.647±.175	.812±.154	—
toybig fresh 4-hop	1.5B	1600	.416±.194	.615±.185	.800±.128	—
umls (real) 3-hop	1.5B	43	.279±.062	.248±.054	.256±.023	.256±.182
umls (real) 2-hop	1.5B	17	.706±.059	.725±.034	.725±.034	.647±.059

Small real graphs. UMLS and CoDEx-S use the main-table protocol at 1.5B and 0.5B. The three 0.5B blocks match the 1.5B blocks in graph, evaluation set, n, evidence, training steps, token limit, batch scaling and absence of warm-up, allowing direct comparison.

Table 22: Small graphs, UMLS: correct-and-valid rate (greedy final) at Qwen2.5-1.5B and -0.5B, no warm-up, batch-normalised advantages. –: not planned. $^{\dagger k}$: k seeds.

Set	n	Block	Model	*CDU*	*COMBINED*	*COU*	*GATED*	*EOG PATH*	*STEP SUM*	*OUTCOME*	*OUTCOME + FMT*
3-hop	70	umlsB	1.5B	.729 ± .196	.871 ± .000	.438 ± .022	.590 ± .243	.186 ± .122	.600 ± .405	.000 ± .000	.219 ± .044
2-hop held-out	24	umlsB	1.5B	.750 ± .182	.847 ± .024	.569 ± .024	.639 ± .173	.222 ± .105	.556 ± .446	.000 ± .000	.306 ± .105
3-hop	70	umlsB05	0.5B	.586 ± .507	.000 ± .000	.457 ± .396	.276 ± .478	.562 ± .079	.000 ± .000	.000 ± .000	.005 ± .008
2-hop held-out	24	umlsB05	0.5B	.514 ± .449	.000 ± .000	.417 ± .361	.250 ± .433	.528 ± .158	.000 ± .000	.000 ± .000	.000 ± .000

Table 23: Small graphs, CoDEx-S: correct-and-valid rate (greedy final) at Qwen2.5-1.5B and -0.5B, no warm-up, batch-normalised advantages. –: not planned. $^{\dagger k}$: k seeds.

Set	n	Block	Model	CDU	COMBINED	COU	GATED	EOG PATH	STEP SUM	OUTCOME	OUTCOME + FMT
3-hop	200	codexsB	1.5B	$.977 \pm .015$	$.958 \pm .029$	$.703 \pm .259$	$.968 \pm .012$	$.813 \pm .148$	$.537 \pm .413$	$.448 \pm .136$	$.518 \pm .445$
2-hop held-out	60	codexsB	1.5B	$.939 \pm .010$	$.950 \pm .033$	$.828 \pm .048$	$.967 \pm .000$	$.744 \pm .202$	$.511 \pm .462$	$.500 \pm .148$	$.544 \pm .457$
3-hop	200	codexsB05	0.5B	$.000 \pm .000$	$.312 \pm .540$	$.000 \pm .000$	$.000 \pm .000$	$.590 \pm .160$	$.000 \pm .000$	$.002 \pm .003$	$.000 \pm .000$
2-hop held-out	60	codexsB05	0.5B	$.000 \pm .000$	$.289 \pm .500$	$.000 \pm .000$	$.000 \pm .000$	$.506 \pm .192$	$.000 \pm .000$	$.000 \pm .000$	$.000 \pm .000$

Table 24: Small graphs, 2Wiki: correct-and-valid rate (greedy final) at Qwen2.5-1.5B and -0.5B, no warm-up, batch-normalised advantages. –: not planned. $^{\dagger k}$: k seeds.

Set	n	Block	Model	CDU	COMBINED	COU	GATED	EOG PATH	EOG RELEASE	STEP SUM	OUTCOME	OUTCOME + FMT
held-out	778	eogB3	1.5B	$.278 \pm .015$	$.132 \pm .229$	$.000 \pm .000$	$.000 \pm .000$	$.000 \pm .000$	$.000^{\dagger 1}$	$.028 \pm .028$	$.000 \pm .000$	$.001 \pm .001$
held-out	778	eogB05	0.5B	$.005 \pm .008$	$.000 \pm .000$	$.000 \pm .000$	$.000 \pm .000$	$.000 \pm .000$	–	$.003 \pm .005$	$.000 \pm .000$	$.000 \pm .000$

G EXTENDED RELATED WORK

Table 25: Related rewards, including exclusions. Signal gives the process input and step check. Ref., zero and cov. denote the reference proof or path information read (excluding the answer key), learning without proof labels and coverage control. The final columns describe our implementations or reasons for exclusion.

work	signal (check)	ref.	zero	cov.	arm	reads, or why not run
controls (Guo et al., 2025; Shao et al., 2024)	the answer	no	yes	no	OUTCOME-ONLY	the answer, the RLVR standard
					OUTCOME WITH FORMAT	the answer, given a parsed claim line
K2V (Yuan et al., 2026b)	LM judge over a checklist (judged)	no	yes	no	GATED K2V-JUDGE	answer + answer $\times$ our process term language-model judge, our criteria
Explore-on-Graph (Yan et al., 2026b)	reference-path recall (overlap)	yes	no	no	REFERENCE-PATH EOG-RELEASE	answer + reference-path recall recall only, released code
VPRM (Pronesti et al., 2026)	step labels (lookup)	yes	no	no	STEP-SUM	answer + share of supported claims
Peak-Then-Collapse (Sun & Kazakov, 2026)	STEP SUM with reference chain (existence)	yes	no	no	none	tool-interface policy, own reward ladder; discussed in §6
KGs as implicit RMs (Kansal & Jha, 2026)	reference paths (overlap)	yes	no	no	none	reference-path class, covered above
CoEvoKG (Li et al., 2026)	path support (co-occurrence)	no	partly	no	none	hyperlink graph, no ontology
CorVer (Fan et al., 2026)	corpus co-occurrence (statistic)	no	yes	no	none	no graph
Lean process RL (Kim & Yun, 2026)	formal checker (decided)	no	yes	fixed	none	formal language, checks submitted formal proofs
Ours	**verifier verdicts (entailment)**	**relations**	**yes**	**yes**	**CDU** **COU**	$R_{\text{out}} + R_{\text{proc}}$**, §3.2** **two-band form, App. A**

Process rewards for reasoning. Step-level supervision began with learned process reward models trained on human step labels (Uesato et al., 2022; Lightman et al., 2024). These models estimate step correctness for sample reranking or policy training. In RLVR, step-level and outcome rewards have been compared for small models on mathematics (Palandye et al., 2026). Prompted language models also judge steps against synthesised checklists in knowledge-intensive domains (Yuan et al., 2026b). Formal-language reasoning permits programmatic checking. Lean serves as a step oracle in GRPO, locating the first failing tactic for credit assignment (Kim & Yun, 2026). Other systems

align natural-language steps with prover-checked formal intermediates and give high reward only to fully validated chains (Chen et al., 2026), or supervise logical steps with a symbolic checker (Xu et al., 2026).

Two findings question whether an explicit process reward is necessary. GRPO with outcome reward acts as an implicit process reward model (Sullivan & Koller, 2026). Outcome reward also improves base models' intermediate reasoning when a language model evaluates it after training (Wen et al., 2026b). Other methods combine both signals within GRPO. PRPO adjusts process-derived token advantages toward outcome advantages because process-only advantages can cause early output truncation (Ding et al., 2026a). VeriGate applies step supervision only when the outcome verifier cannot separate a group (Agrawal et al., 2026). RLVP adds verifiable path penalties to outcome reward, matching our total reward's additive form (Li & Shi, 2026). Learned natural-language proof judges continue to improve (Yang et al., 2026), but their bias toward leniency or strictness can be altered (Zhou et al., 2026b). Checking entailment directly avoids this source of judgment error.

Outside formal languages, these methods judge step correctness with a learned or prompted model whose error rate is not known in advance. Formal checkers decide every proof in their languages but require reasoning in their own languages. Neither approach measures during training how often a correct answer has an invalid proof. Existing audits measure this at evaluation using automated theorem provers (Zheng et al., 2026) or inspections of verifier gaming in trained policies (Helff et al., 2026). Our reasoner checks cited claims against graph closure for natural-language reasoning over supplied facts. Verifier coverage can be varied, and a deterministic checker tracks correct answers with invalid proofs at every evaluation checkpoint.

Process rewards over knowledge graphs. Several knowledge-graph reasoning rewards use a graph or reference path. Explore-on-Graph rewards the fraction of reference-path triples appearing as substrings in the trace (Yan et al., 2026b). Verifiable process reward models use reference step labels in medical risk-of-bias assessment (Pronesti et al., 2026). Knowledge graphs as implicit reward models use reference reasoning paths in medical question answering (Kansal & Jha, 2026). Knowledge-to-Verification gates a language-model judge's checklist pass rate on answer correctness, without showing the judge the graph (Yuan et al., 2026b). Search-on-Graph-R1 uses outcome reward over a live Freebase endpoint. It argues that static-subgraph accuracy is limited by retrieval outside the model's control (Sun et al., 2026). Peak-Then-Collapse combines step reward and a reference-chain term over a Freebase tool interface, finding that denser rewards change failure modes without eliminating them (Sun & Kazakov, 2026). CoEvoKG measures lexical path support in the Wikipedia hyperlink graph (Li et al., 2026), while CorVer uses corpus co-occurrence for text question answering (Fan et al., 2026).

Reasoning on Graphs provides the WebQSP and CWQ subgraphs used here. It plans a relation path, grounds it in the graph and answers from retrieved paths, so faithfulness depends on retrieval rather than verification of the written reasoning (Luo et al., 2024). KG-Hopper trains compact models with outcome rewards to traverse graphs with backtracking within one thinking stage (Wang & Yu, 2026). Tree-structured GRPO provides look-ahead feedback at each step for temporal question answering (Wen et al., 2026a). StepGap uses a hybrid entailment and language-model checker to classify multi-hop steps as contradicted, irrelevant or missing a bridge, then uses these verdicts as rewards (Ji et al., 2026). It checks retrieved text rather than graph closure.

Explore-on-Graph, verifiable process reward models, implicit reward models and Peak-Then-Collapse's chain term require a reference path or trace for each question. Explore-on-Graph, CoEvoKG, CorVer and Peak-Then-Collapse check text overlap or triple existence rather than entailment. They therefore do not distinguish a mentioned fact from a cited step or credit derived facts. Knowledge-to-Verification's judge receives no graph. Methods requiring reference paths cannot run without those labels. Our verifier checks each cited claim for membership in OWL 2 RL closure, crediting derived facts and rejecting unentailed claims. Contracts are the reference path's relation sequence widened by the ontology, and one verifier serves nine graphs. The reward requires no reference trace, path overlap or judge. This permits learning without proof labels, and the largest gap to outcome reward is on WebQSP, which has no warm-up (§6). Re-implementing published shapes with our verifier compares reward shapes under one checker, and the implementation names identify these substitutions. We also compare with the negative CWQ result of Sun & Kazakov (2026). That

study uses triple existence and a reference-chain term, a tool-interface policy, and its own sequence of reward designs. None checks steps against closure.

Verifier quality and reward hacking. Studies of verifier error find that moderate independent noise in an outcome verifier can be tolerated (Plesner et al., 2026). Systematic errors can cause plateaus or collapse where random errors at the same rate do not (Egashira et al., 2026). Learning also exhibits a phase transition governed by the difference between true-positive and false-positive rates (Rad et al., 2026). Dense rewards from misaligned oracles can perform worse than sparse rewards (Yuan et al., 2026a). Leaky test suites reward false positives (Zhang, 2026), and spurious rewards can activate memorisation shortcuts (Yan et al., 2026a). PAC-Bayesian bounds relate process reward model error to the accuracy of the formal verifier that labels its training data (Rahman & Alsharari, 2026). Under incomplete graph evidence, no hard acceptance rule avoids both false accepts and false rejects (Li & Song, 2026).

Other work treats the verifier as a noisy reward channel and corrects its errors, for example by correcting noise in Monte Carlo step labels during training (Xie et al., 2026). Re-verification of spurious-reward datasets finds clean-data contamination and shows that incorrect annotations reduce accuracy (Zhu & Kang, 2026). Verifier audits find that widely used mathematics checkers fail on identical inputs for different reasons (Xin, 2026), fuzzing exposes faults that policies could exploit (Ray, 2026), and exact-match false-negative rates vary by language (Zhou et al., 2026a), demonstrating error correlated with input attributes.

These studies vary answer-level errors, scalar accuracy or synthetic oracles. None varies coverage of a real step-level verifier by a specified amount and measures the resulting policy reasoning. We mask closure edges, identifying exactly which facts the verifier cannot access. Reduced coverage lowers validity on synthetic graphs and by .018 on Dynamic-KGQA, where abstention does not recover it (§6.1). The incompleteness result of Li & Song (2026) motivates this coverage control: incomplete evidence requires errors in at least one direction, and the verification policy determines how uncertainty is handled.

Reward design and padding. Process-reward ordering has received less attention than reward accuracy. Quality-diversity stress tests show that adding steps can raise the mean of learned step scores but cannot raise their minimum (Shihab & Afrin, 2026). This concerns aggregation of a learned scorer. Among the methods compared, none provides an ordering framework for verified steps, penalty for supported but unused facts, or account of which orderings a weighted sum can satisfy. §3.2 and App. A define five properties, give a two-band reward that satisfies them but collapses in training, and measure padding under our weighted sum.

Correct answers with unfaithful reasoning. A correct answer unsupported by its written reasoning is studied at evaluation as a faithfulness problem. Sufficiency, completeness and necessity diagnostics assess whether answer-relevant information passes through the chain of thought and can guide training (Jia et al., 2026). In multimodal RLVR, a consistency reward reduces a persistent gap between reasoning and answers (Cao et al., 2026). Planners can also receive reward for improving a frozen executor's performance, since answer correctness alone does not establish trace usefulness (Han et al., 2026). These methods measure or reward dependence on reasoning rather than checking steps against a knowledge base.

Verify-then-repair at inference. SR-RAG builds a symbolic graph during inference, checks entailment with Prolog, and returns failure traces so the language model can add facts (Wang et al., 2026). Code-on-Graph regenerates code after execution failure or empty results (Ding et al., 2026b). GRACE returns evidence-support rejections to retrieval for a fixed open-source model (Zhao et al., 2026). These systems check model-built graphs or code execution rather than cited steps against a fixed graph's closure. We use an existing repair loop to test whether the verifier used in training also improves frontier models. Two feedback turns increase correct-and-valid rates on three graphs (§6.1).

Scope of the comparison. Among the compared methods, none checks each step of a natural-language proof against graph closure without a reference trace or a judge, none provides configurable coverage for measuring the effect of incomplete verification, and none specifies which order-

ings weighted sums can satisfy. We know of no evaluation of learning without proof labels under entailment checking.