CAP-7: BINDING-AWARE AND COMMITMENT-COMPLETE VERIFICATION FOR TASK-LEVEL AFFORDANCE REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Task-level affordance reasoning requires vision-language models to turn images and manipulation instructions into plans that connect objects, actions, states, and motions. Yet evaluating these elements separately leaves two failures unresolved: plausible elements may be connected incorrectly, which we term *Binding Blindness*, and required steps may be omitted without being penalized, which we term *Omission Blindness*. To expose incorrect bindings, we introduce CAP-7, a typed program representation linking objects and roles to actions, states, dependencies, and motions. We construct CAP-7-PD for supervision and a 20k-program extension for community research. Controlled changes to roles, action order, and motion paths test whether the resulting plans preserve the relations required by the task. To detect omissions, we develop commitment-complete verification that retains missing requirements in the score and penalizes unsupported additions. Combined with a check of whether the submitted action order executes legally, this verification provides a training reward. Rescoring unchanged model outputs reveals misleading gains under the original verifier. Under matched training conditions, replacing only the reward raises the mean share of sampled orders judged legal by an independent symbolic engine from 51.3% to 94.4%, compared with 85.2% for the shared supervised initialization; unmatched step pairs also decrease. These results show that complete task verification can improve the structure and symbolic executability of learned manipulation plans. `https://anonymous.4open.science/r/CAP7-EBF5/`.

1 INTRODUCTION

Task-level affordance reasoning requires vision-language models (VLMs) to turn visual observations and manipulation instructions into plans that connect objects, actions, states, and motions. Recent VLMs can locate task-relevant objects, propose action sequences, and refine long-horizon behavior using visual or physical feedback (Ni et al., 2024; Yang et al., 2025; Mon-Williams et al., 2025). These capabilities provide the elements of a manipulation plan, but execution also depends on how they are connected: which tool acts on which target, which state enables each step, which actions must precede others, and which motion paths are feasible (Figure 1). A plan may therefore pass checks on individual elements while connecting them incorrectly. This raises the central evaluation question: **Can we evaluate a plan's relations, rather than its elements in isolation?**

Existing methods enrich manipulation plans with affordance cues, sequential structure, programmatic actions, and feedback (Yu et al., 2025a; Mu et al., 2024; Huang et al., 2025a; Zhao et al., 2025). Yet a plan may bind an action to the wrong object or state, or specify dependencies and motions that conflict with the rest of the task. We call these incorrect connections ***Binding Blindness***. A second failure arises in evaluation: when a verifier scores only the relations a model expresses, required steps and relations that the model omits may escape assessment. We call this ***Omission Blindness***. Indeed, figure 2 illustrates the distinction: a model can be accurate about the relations it states while covering only a small fraction of those the task requires.

To make both failures measurable, we first represent the task's connections explicitly. CAP-7 encodes a manipulation plan as a typed program whose shared identifiers link objects and roles to steps,

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

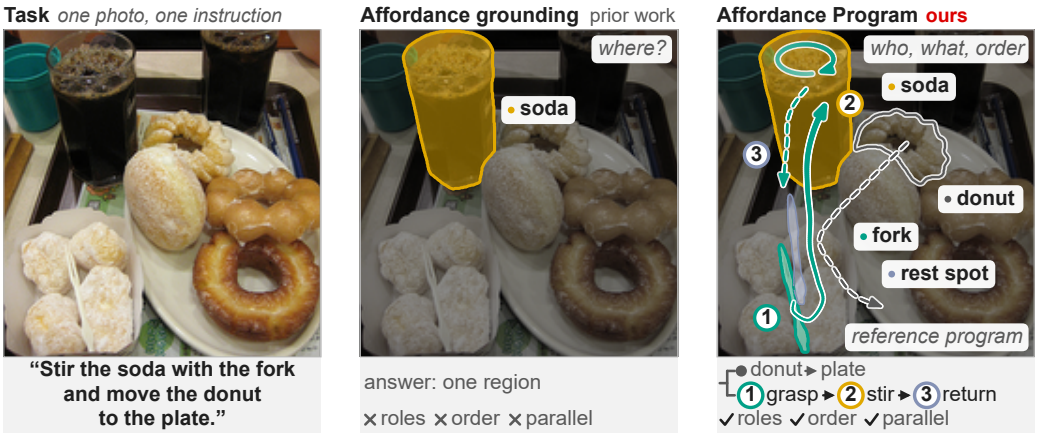


Figure 1: **A task requires relations between predictions.** From one photograph and one instruction, a usable plan must assign object roles, order the actions, preserve parallel branches, and specify the motion path. A field-only affordance answer can identify a region while leaving these relations unresolved.

dependencies, states, and motions. CAP-7-PD then provides program-level supervision for learning these connections across task families. With the relations specified, we can test their correctness by changing role assignments, required order, or motion paths while keeping individual predictions plausible. Figure 3 shows that such changes can break the task even when its individual elements appear correct. The program representation also reveals which required connections are absent from a prediction. The remaining challenge is to ensure that these omissions affect the score.

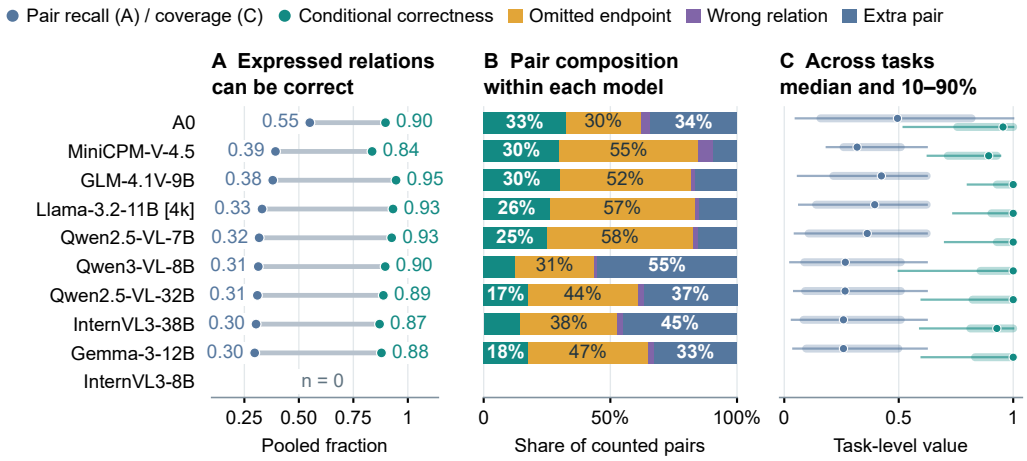


Figure 2: **Models can get the relations they state right while omitting much of the task.** (A) Blue shows how many required step pairs a model includes; teal shows how often its stated relations are correct. A0 and the external models are evaluated on the same tasks, using each model's parseable outputs (Appendix E.2). (B) These outputs contain correct pairs, pairs missing because a step was omitted, wrong relations, and extra pairs. (C) Across tasks, pair coverage also counts extra pairs in its denominator (Eq. 3). Dots mark medians; lines span the 10th–90th percentiles across tasks.

A verifier that scores only emitted relations can miss an omitted step because its required relations disappear from evaluation with it. Likewise, an unmatched step can add unsupported content without receiving an adequate penalty. These scoring failures occurred under the original CAP-7 verifier: scores could increase even as the correctness of expressed relations declined, while sparse and padded programs exposed related problems (Figure 4). Task-level verification must therefore account for both requirements left unstated and content added without support.

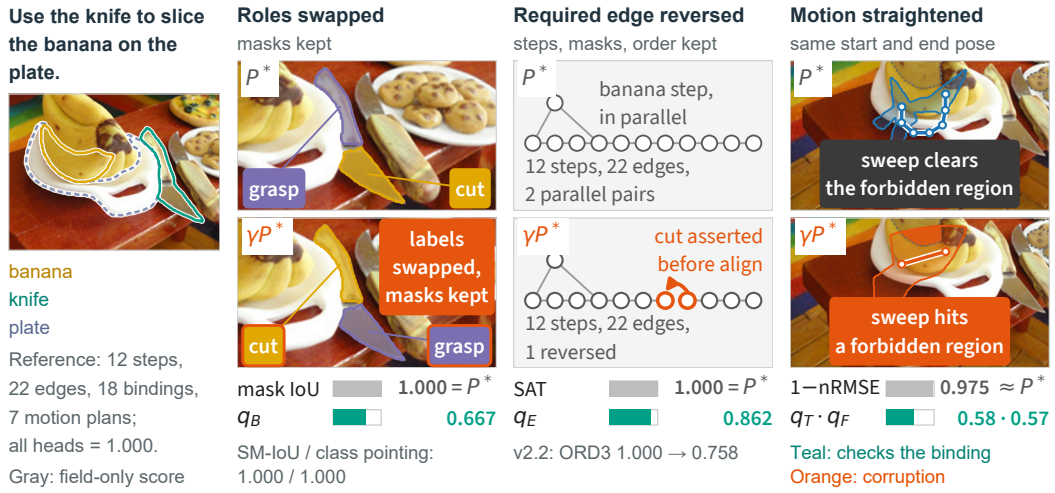


Figure 3: **A plan becomes testable when its links are explicit.** Edits to one reference preserve selected fields but alter roles, order, or motion. Binding (q_B), order (q_E), trajectory (q_T), and sweep (q_F) checks respond where field-only scores can fail. SAT checks required edges; ORD3 checks all step pairs; v2.2 denotes complete verification.

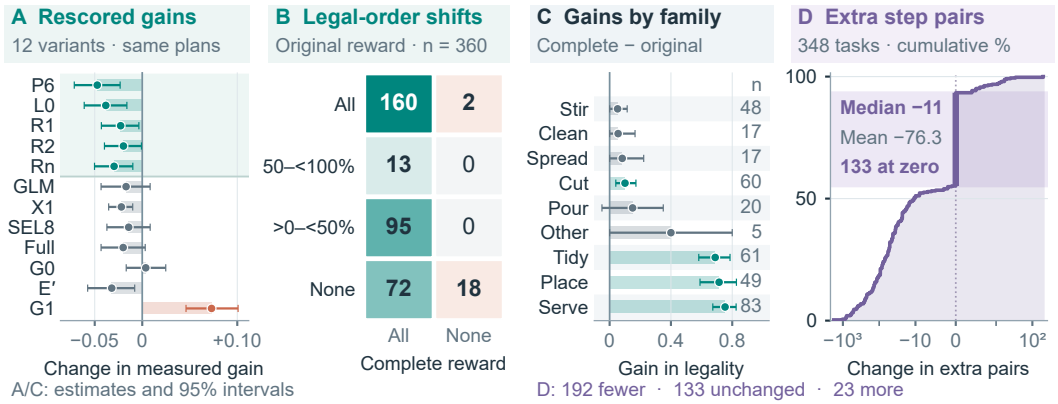


Figure 4: **Complete verification changes measured gains and learned plans.** (A) Rescoring the same plans reduces gains over the supervised start for reward training (P6), replay gating (L0), and selected fine-tuning (R1/R2/Rn; teal); coral marks span-weighted supervised continuation (G1). Other variants: Appendix F. (B–D) Replacing only the reward, with a shared start, procedure, and 32-update budget, improves legality (B), increases mean legality across nine families (C), and reduces extra pairs involving unmatched steps (D). B counts 360 paired tasks by the fraction of 50 sampled orders that execute legally; All/None mean 100%/0%. D uses 348 jointly parseable tasks; the shaded band marks 133 with no change. A/C show 95% task-bootstrap intervals from 10,000 resamples; teal in C marks intervals above zero.

To address these verification failures, we develop a commitment-complete verifier that measures both the correctness of expressed relations and the coverage of required ones, while penalizing unsupported additions. We then check whether the submitted action order executes legally and use the combined result as a training reward. To test whether this change affects what the model learns, we hold the supervised starting model, training procedure, and update budget fixed and replace only the reward. The mean share of sampled orders judged legal by an independent symbolic engine rises from 51.3% to 94.4%, compared with 85.2% for the shared supervised initialization; unmatched step pairs also decrease (Figure 4). Together, these results support three contributions: (i) CAP-7, a typed representation that makes task relations jointly checkable; (ii) CAP-7-PD for program-level supervision and CAP-7-PD-20k, a 20k-program extension for community research on task-level affordances; and (iii) commitment-complete verification with an execution-aware reward that carries the task's requirements into learning, including requirements absent from a model's output.

2 Related Work

Vision-language reasoning and embodied planning. Recent models ground instructions in task-relevant parts (Wan et al., 2025) and spatial targets (Yuan et al., 2025; Zhou et al., 2025; Song et al., 2025), then extend planning to sequential affordances (Yu et al., 2025a), coupled affordance and trajectory prediction (Ji et al., 2025), and parallel task dependencies (Duan et al., 2026). Programmatic planners express actions and geometric constraints (Mu et al., 2024; Huang et al., 2025c); scene-graph systems connect visual entities, relations, and states to planning (Huang et al., 2025b; Ju et al., 2026). These advances supply increasingly structured plans, but the structure alone does not establish that a predicted program includes and satisfies its task requirements. We therefore connect roles, dependencies, state providers, and motions through shared identifiers, so CAP-7 can check their agreement within the same program.

Structured evaluation and learning from verification. Relation-aware benchmarks already assess caption coverage and binding (Lu et al., 2025), temporal associations (Saravanan et al., 2025), and missing or extra planning steps (Li et al., 2024). Reward signals assess visual accuracy (Liu et al., 2025; Tan et al., 2025), instruction checklists (Viswanathan et al., 2025), and agreement with reference skills or action prefixes (Bo et al., 2025; Wu et al., 2026). Process supervision (Lightman et al., 2024; Luo et al., 2025) and perception-focused feedback (Min et al., 2026; Hou et al., 2026) further refine what receives credit. Our question is how to preserve required task relations in the score when their endpoints are omitted, while also charging unsupported additions. We derive this accounting for connected programs and combine it with state-supported execution checks, then test both the measured gains and the programs learned from the resulting reward.

3 CAP-7: From Fields to Task Programs

3.1 Representing relations explicitly

Learned planners describe preconditions and effects (Liang et al., 2025; Huang et al., 2026); we link these to object roles and motion within one typed program:

$$P = (\mathcal{O}, \mathcal{S}, E, L, \mathcal{B}, \Sigma, \mathcal{T}). \tag{1}$$

Here $\mathcal{O}$ records objects and roles, $\mathcal{S}$ steps, E necessary dependencies, L a proposed execution order, $\mathcal{B}$ role–region bindings, Σ states and their providers, and $\mathcal{T}$ motion plans. Shared identifiers connect a grasped object, its state, and later motion. Following parallel planning (Duan et al., 2026), E specifies necessary precedence, while L proposes an execution order to check. This preserves independent actions (Appendix A.3).

3.2 Matching once, checking seven requirements

Joint verification requires consistent step correspondences. Building on skill matching (Bo et al., 2025), we compute a maximum-weight partial one-to-one assignment μ from action, moving entity, and position compatibility. Seven heads check distinct requirements, sharing this assignment wherever steps are involved (Table 1). Path similarity and geometric feasibility are separate because a nearly unchanged path can cross an obstacle (Figure 3).

Table 1: **One program, seven complementary checks.** Step-dependent heads share μ. The checks cover symbolic relations and image-plane geometry.

Head	Requirement checked
q_O: objects	Object identity and task role, including annotated alternatives.
q_S: steps	Action, moving entity, and feasible position of each step.
q_B: bindings	The object, role, part, and region assigned to a matched step.
q_E: order	Reference precedence and pairwise dependence or independence.
q_C: states	Preconditions, their providers, and the effects added or deleted.
q_T: trajectories	The path assigned to each moving step.
q_F: geometry	Bounds, obstacles, target proximity, and contact along the path.

3.3 Learning the connected program

To learn these relations together, we construct the CAP-7-PD dataset. Grounded intermediate supervision connects reasoning to actions or visual points (Zawalski et al., 2025; Ni et al., 2025); our records link each action's roles, dependencies, state providers, and motions. Models propose and adjudicate records; structural checks determine acceptance (Table 2; Appendix A.2). Training from fragments to full programs yields the jointly supervised 8B model (A0), the shared start for reward training.

Table 2: **CAP-7-PD used in this work.** Counts are task programs, before conversion into training examples. The nine task families have image-disjoint splits.

	Training	Calibration	Test	Total
Programs	1,254	84	**360**	**1,698**

4 Keeping Task Requirements in the Score

Explicit relations can still escape evaluation if deleting a step also deletes the relations being checked. Existing planners check feasibility or repair symbolic rules (Yang et al., 2025; Hao et al., 2026), and rewards compare skills or sequence lengths (Bo et al., 2025; Tan et al., 2025). We instead make required relations remain countable after omission. We first define what receives credit, then separate missing from incorrect content, and finally require rewarded programs to admit a valid execution order (Figure 5).

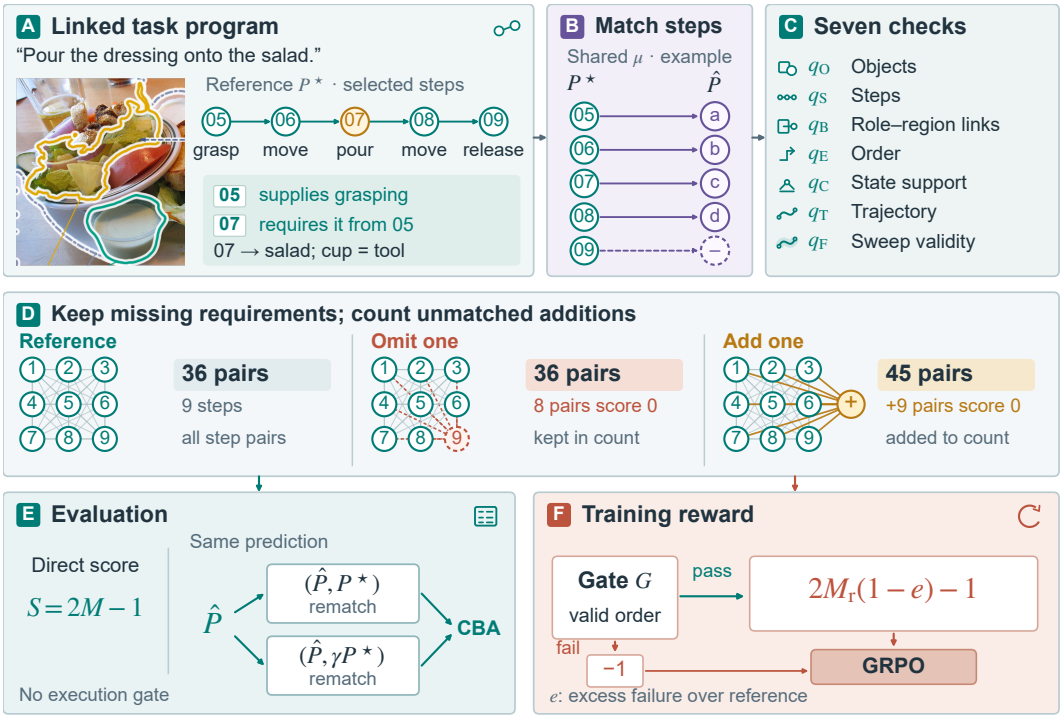


Figure 5: **Missing steps must not erase the relations we check.** (A–C) A reference pouring task links roles, actions, and states; shared step matching supports seven checks. (D) With matching and retained-step order fixed, omitting one of nine steps leaves all 36 pairs in the score, with eight receiving zero; an unmatched addition raises the count to 45. Lines denote counted pairs, not dependencies. (E–F) Evaluation uses direct scores or corruption-pair binding accuracy (CBA) against intact and altered references. Training additionally requires a valid submitted order and penalizes replay failures above the reference level.

4.1 Count both what is required and what is written

Scoring only predicted step pairs rewards omission; scoring only reference pairs overlooks additions. We therefore count both required pairs and extra pairs. Let $U^\star$ contain all unordered pairs of physical

reference steps. Dependency closure, including controller nodes, classifies each pair as forward, reverse, or independent. A pair earns credit only if both steps appear in the predicted order $\hat{L}$ and their relation is correct. Let m count steps in $\hat{L}$ matched to physical reference steps and $u = |\hat{L}| - m$. Three-way order agreement (ORD3) is

$$D = |U^\star| + um + \tbinom{u}{2}, \qquad \mathrm{ORD3} = \frac{N_{\mathrm{correct}}}{D}, \qquad q_E = H_{0.02}(\mathrm{SAT}, \mathrm{ORD3}). \tag{2}$$

The denominator retains missing reference pairs and adds every pair involving an unmatched step. SAT checks whether $\hat{L}$ respects reference dependencies, counting missing endpoints as failures. The order head q_E combines these two checks using a smoothed harmonic mean $H_{0.02}$ (Appendix B.1).

We apply the same principle to states and motion. State precision counts all written preconditions and effects; recall checks required states and the steps supplying them. When the reference requires motion, unmatched trajectories also enter the denominators. Thus four heads change—order, states, trajectories, and geometry—while the other three remain unchanged. Counting additions and retaining omissions are the first two completeness conditions, C1 and C2. With matching and correct contributions fixed, an extra charged element changes c/d to $c/(d+k) \le c/d$; Appendix B.2 gives the full conditions.

4.2 Distinguish missing content from incorrect content

A low score may reflect missing steps, wrong relations, or both. We make that distinction explicit by factoring order agreement. If $N_{\mathrm{expressed}}$ counts reference pairs with both endpoints matched into $\hat{L}$, then

$$\mathrm{ORD3} = \underbrace{\frac{N_{\mathrm{expressed}}}{D}}_{\text{pair coverage}} \times \underbrace{\frac{N_{\mathrm{correct}}}{N_{\mathrm{expressed}}}}_{\text{conditional correctness}}. \tag{3}$$

Coverage measures how much required structure the program expresses; conditional correctness measures how accurately it relates those steps. Their product distinguishes accurate partial plans from complete ones. The identity holds for expressed-pair programs or pooled counts; unlike reference-pair recall, coverage also charges extra pairs.

We evaluate programs in two ways. The direct score $S = 2M - 1$ averages the applicable heads in M. Contrast accuracy—corruption-pair binding accuracy (CBA)—asks whether the same prediction scores better against the reference than against a deliberately altered reference. Steps are rematched; wins, ties, and losses receive $1, \frac{1}{2}, 0$, averaged within each applicable family, across applicable families, then across tasks. Such evaluations depend on their contrasts (Huang et al., 2024; Kamath et al., 2024): modification-only contrasts barely distinguish an edgeless plan, even with complete heads (Table 3). We test omissions separately with deletion contrasts. Rescoring keeps the corruption families fixed: the original rule uses a strict margin, whereas the complete rule adds tie tolerance and a head-specific floor (Appendix B.3).

Table 3: **Each design choice addresses a specific scoring failure.** Offline comparisons test constructed programs, fixed candidates, and reference execution. Bold marks the targeted outcome. Appendices B and G.3 give full results; the combined training intervention is evaluated separately.

Design question	**Comparison**	**Observation**
Do omissions enter the direct score?	Edgeless program: original → complete	S: $0.862 \rightarrow \mathbf{0.676}$; reference 1
Do contrasts expose omissions?	Same complete heads: four families → add deletion	Edgeless CBA: $0.998 \rightarrow \mathbf{0.874}$
Does aggregation hide order changes?	Power mean $p = -1$ → arithmetic mean	Order-different tie rate: $0.611 \rightarrow \mathbf{0.000}$
Must every permitted order execute?	Submitted-order witness → all-sampled-order gate	References rejected: $\mathbf{0/360} \rightarrow 21/360$
Does replay penalize the reference?	Charge failure above the reference level	Reference self-score: $\mathbf{1}$ on $360/360$

First two rows: means over 360 constructed programs; complete-protocol reference CBA is 0.9995/0.9997. Aggregation uses original heads on 2,835 gated candidates across 360 tasks; ties concern 5,834 order-different pairs. Replay comparisons sample 20 orders per reference.

4.3 Reward a valid order without overconstraining the task

A program can describe the required relations yet propose an order that cannot execute. We apply the graded reward only if structural checks pass, physical steps have states, the graph is acyclic, and the submitted order respects dependencies and executes under its declared states; otherwise it receives -1. Requiring every sampled order to execute would be too strong: independent task actions can still compete for one hand. The submitted-order check preserves legal reference plans that an all-orders gate rejects (Table 3).

Passing this check should not excuse a program that permits many failing orders. We use reference failure as the baseline for this penalty. Let $\widehat{\ell}(P)$ be the failure fraction over 20 orders sampled by the ready-node procedure, M_{r} the mean complete-head score over the reference-fixed active set, and G the combined structural and submitted-order verdict:

$$e = [\widehat{\ell}(\hat{P}) - \widehat{\ell}(P^{\star})]_{+}, \qquad R_{\mathrm{cc}} = \begin{cases} -1, & G = 0, \\ \mathrm{clip}(2M_{\mathrm{r}}(1-e) - 1, -1, 1), & G = 1. \end{cases} \tag{4}$$

Subtracting reference failure makes its excess penalty zero, the third completeness condition (C3). We use an arithmetic head mean because it distinguishes order changes that a harmonic power mean can obscure (Table 3). This reward replaces the original gated head mean in GRPO (Shao et al., 2024). Contrast accuracy and the independent execution engine remain evaluation measures; neither supplies the reward. Algorithm 1 specifies the procedure.

5 Experiments

Our experiments test the design chain: joint supervision should improve relational agreement; complete verification should distinguish score gains from better relations; and the reward should change the structure and execution of learned programs.

Setup. The jointly supervised model is the common reference for representation, backbone, and reward comparisons. We evaluate 360 tasks using paired bootstrap intervals. Contrast accuracy uses instance, order, and composite corruptions excluded from original optimization; parse failures receive 0.5. Relation-score differences use jointly parsed programs, whereas all-task execution legality assigns parse failures zero. Calibration and development data select checkpoints. Appendix D specifies models, prompts, candidate pools, and sampling.

5.1 Program supervision improves relational agreement

Program supervision improves overall relation agreement at similar pair coverage. External models often relate their expressed steps correctly while leaving reference content uncovered (Figure 2). In the cake example, both external programs replace piercing with cutting: their matched steps are correctly ordered, but the reference plan remains incomplete. Our jointly supervised model retains the reference steps and their ordering relations before any reward training (Figure 6).

Table 4: **Program supervision improves relations at similar coverage.** Program supervision reports levels; controls report changes from it with 95% paired intervals. Coverage uses pooled pair counts; ORD3 uses task means and task-paired differences. Bold links similar coverage to the program–sequence score differences.

Model	Training target	Pair coverage	Order / Δ	Contrast / Δ
A0	Program	**0.362**	0.465	0.828
N-1	Flat sequence	**0.357**	**−0.025** [−0.043, −0.008]	**−0.036** [−0.055, −0.018]
B-0	Program, second backbone	0.352	−0.002 [−0.017, +0.015]	−0.001 [−0.019, +0.016]

Order: three-way agreement (ORD3); contrast: corruption-pair binding accuracy (CBA). A0/N-1/B-0 correspond to the three rows. Coverage: 333/335/328 parsed outputs; ORD3: jointly parsed pairs; CBA: all 360 tasks; 5,000 bootstrap resamples. A0 and N-1 share the three-epoch SFT-1 checkpoint and two-epoch SFT-2 budget (314 updates); N-1 learns flat sequences with template lifting. B-0 applies the program recipe to Qwen3-VL-8B.

From the same fragment-supervised checkpoint and with matched second-stage updates, program supervision improves order agreement and contrast accuracy over flat sequences at similar coverage (Table 4). A second 8B backbone gives comparable program-learning results (Appendix E).

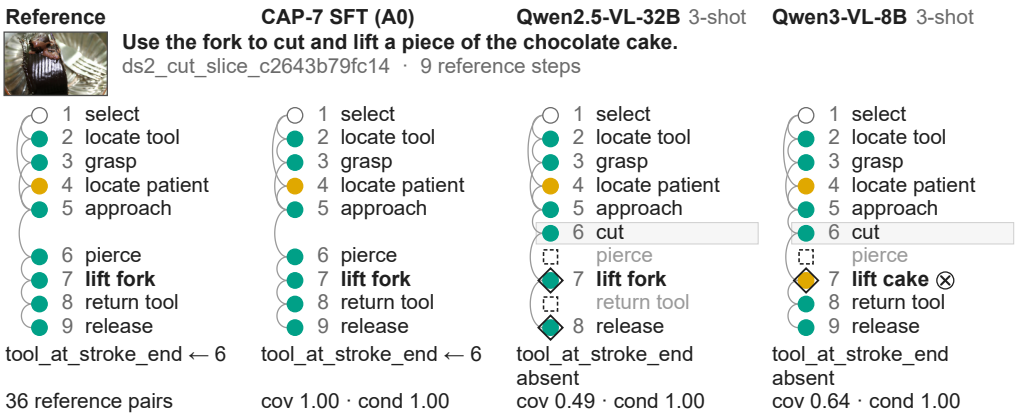


Figure 6: **Joint supervision learns a connected task program.** Our supervised model (A0) expresses all nine reference steps. External models correctly relate their matched steps while omitting reference content. Dashed slots mark missing steps; shading marks unmatched additions, diamonds differing state providers, and ⊗ wrong bindings. Arcs show dependencies; teal/yellow nodes denote tool/target roles. The coverage and correctness below each program distinguish these outcomes (protocols: Appendix D.1).

5.2 Higher scores can accompany worse relations

Proxy optimization can diverge from the intended outcome (Khalaf et al., 2025). Both direct scorers penalize every constructed defect family on average, yet modification-only contrasts barely distinguish an edgeless program from the reference. Complete accounting strengthens the direct penalty; deletion contrasts test the missing relations (Table 3; Appendix B).

Under complete verification, the same programs from original-reward training no longer show a statistically detectable contrast-accuracy gain over the supervised start (Figure 4A). Across five training variants, original contrast accuracy rises while correctness on expressed relations falls (Figure 7B). Reward training adds unmatched content; selection-based fine-tuning produces sparser programs. Some gains survive, including dependency reconstruction and replay-gated training (Appendix F). These results distinguish a higher contrast score from more accurate relations.

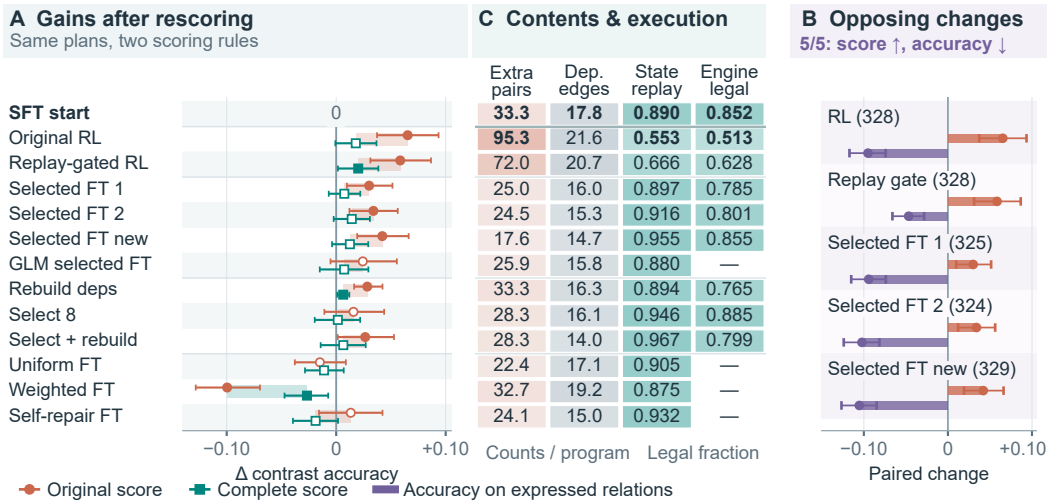


Figure 7: **A higher score can accompany less accurate relations.** (A) Rescoring the same programs changes their gains over the supervised start; bands connect the two estimates. (B) All five trained variants gain original contrast accuracy but lose correctness on expressed relations. (C) Aligned cells link these changes to program content and execution: counts use parsed programs; state replay and engine legality use all 360 tasks. Whiskers: paired 95% task-bootstrap intervals (score gains: 360 tasks, 5,000 resamples; expressed-relation correctness: n in B, 10,000). Appendix F defines the variants and comparison sets.

5.3 CHANGING THE REWARD CHANGES THE LEARNED PROGRAM

We next replace the reward while holding the supervised initialization, training seed, and optimization settings fixed. Both selected checkpoints have the same update budget. The complete reward curbs extra step pairs during training and improves execution legality (Figure 8). A separate symbolic engine evaluates the resulting programs using reference initial states and fixed action rules, independently of states supplied by the prediction. Appendix D.1 gives the run and checkpoint-selection settings.

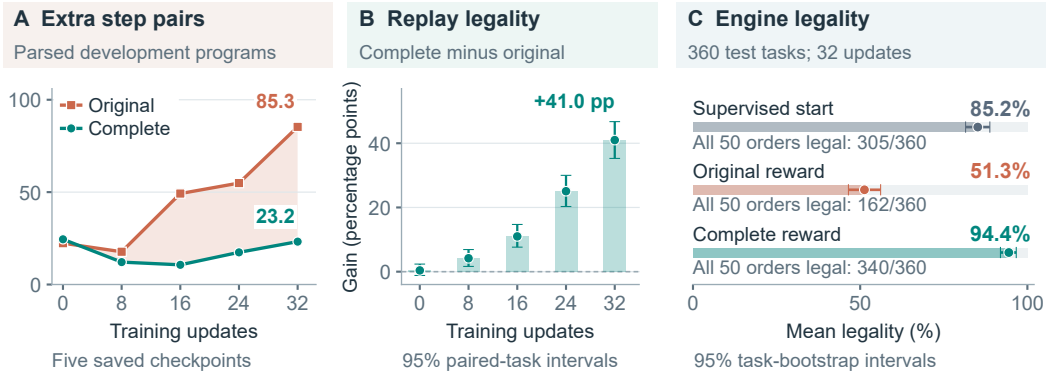


Figure 8: **Complete rewards curb padding and improve execution legality.** (A) Extra step pairs accumulate under the original reward after early training; shading marks the gap between checkpoint means. (B) The paired gain in declared-state replay legality increases during training. (C) Mean symbolic-engine legality and task counts with all 50 sampled orders legal. A averages parseable development programs; B/C use all 254/360 tasks. Whiskers: 95% task-bootstrap intervals (B: 5,000 paired resamples; C: 10,000). The training seed (42) and budget are held fixed across reward conditions. Appendix G gives full protocols and outcomes.

The gain concerns execution legality: task-goal attainment and held-out contrast accuracy show no detectable improvement, while the step-deletion contrast score falls. Appendix G gives paired comparisons for all outcomes.

Replacing the reward changes the learned programs: the model produces fewer unmatched step pairs and achieves higher symbolic execution legality than both the original-reward model and the shared supervised starting point (Figure 8). The gain over the supervised start persists on tasks where both outputs are parseable, establishing an improvement beyond formatting. Together, results across training seeds show that task-level verification guides models to produce programs that better satisfy execution constraints (Appendix G).

6 DISCUSSION AND LIMITATIONS

Success on separate skills need not survive their composition (Li et al., 2025). CAP-7 separates task coverage, relational correctness, and execution legality, making their different responses to optimization visible. The reward intervention changes both program structure and symbolic execution legality, showing how the task requirements checked by a verifier can shape the plans a model learns to produce. This connection motivates integrating task-level verification with perception and control during physical robot execution.

7 CONCLUSION

We introduce CAP-7 to make the relations among objects, actions, and states explicit and verifiable, and CAP-7-PD to provide joint program supervision. Our commitment-complete verifier accounts for omitted requirements and unsupported additions; an execution-aware reward carries these checks into learning. Replacing the reward reduces padding and raises symbolic execution legality above the supervised starting point. Together, these contributions connect task specification, verification, and learning. The broader design principle is to build learning signals around what a task requires, including what a model leaves unstated, so that verification actively guides the plans it learns to produce.

AI USAGE DISCLOSURE

AI tools assisted manuscript writing and language editing, including improvements to clarity, grammar, and consistency of expression. Large language models also supported the construction of CAP-7-PD by proposing structured task programs and reviewing their annotations. The dataset construction and validation procedure is described in Section 3.3 and Appendix A.2.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-VL Technical Report, 2025a. URL https://arxiv.org/abs/2511.21631. arXiv:2511.21631.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-VL Technical Report, 2025b. URL https://arxiv.org/abs/2502.13923. arXiv:2502.13923.

Zitong Bo, Yue Hu, Jinming Ma, Mingliang Zhou, Junhui Yin, Yachen Kang, Yuqi Liu, Tong Wu, Diyun Xiang, and Hao Chen. Reinforced Embodied Planning with Verifiable Reward for Real-World Robotic Manipulation, 2025. URL https://arxiv.org/abs/2509.25852.

Shiying Duan, Pei Ren, Nanxiang Jiang, Zhengping Che, Jian Tang, Zhaoxin Fan, Yifan Sun, and Wenjun Wu. RoboPARA: Dual-Arm Robot Planning with Parallel Allocation and Recomposition Across Tasks. In *International Conference on Learning Representations*, volume 2026, pp. 143933–143974, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/e8aa3f5e8b5f24b830f31fd828e2a2a3-Abstract-Conference.html.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 Technical Report, 2025. URL https://arxiv.org/abs/2503.19786. arXiv:2503.19786.

Yilun Hao, Yongchao Chen, Chuchu Fan, and Yang Zhang. Simulation to Rules: A Dual-VLM Framework for Formal Visual Planning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=7tlLpQpGlx.

Xuege Hou, Wenshuo Li, Yali Li, Han Shu, Yuan Wang, Xinghao Chen, and Shengjin Wang. VES-RFT: Rewarding Visual Evidence Sensitivity to Mitigate Hallucinations in Large Vision-Language Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4168–4177, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Hou_VES-RFT_Rewarding_Visual_Evidence_Sensitivity_to_Mitigate_Hallucinations_in_Large_CVPR_2026_paper.html.

Chi-Pin Huang, Yueh-Hua Wu, Min-Hung Chen, Frank Wang, and Fu-En Yang. ThinkAct: Vision-Language-Action Reasoning via Reinforced Visual Latent Planning. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 82782–82802. Curran Associates, Inc., 2025a. doi: 10.52202/085713-2773. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/7749f9c0d5ff109231be21e910a3ced2-Abstract-Conference.html.

Irene Huang, Wei Lin, M. Jehanzeb Mirza, Jacob A. Hansen, Sivan Doveh, Victor Ion Butoi, Roei Herzig, Assaf Arbelle, Hilde Kuehne, Trevor Darrell, Chuang Gan, Aude Oliva, Rogerio Feris, and Leonid Karlinsky. ConMe: Rethinking evaluation of compositional reasoning for modern VLMs. In *Advances in Neural Information Processing Systems*, 2024. doi: 10.52202/079017-0721. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/28aad3b3b315d86910d7f4ee2867dfa4-Abstract-Datasets_and_Benchmarks_Track.html.

Jiani Huang, Amish Sethi, Matthew Kuo, Mayank Keoliya, Neelay Velingker, JungHo Jung, Ser Nam Lim, Ziyang Li, and Mayur Naik. ESCA: Contextualizing embodied agents

via scene-graph generation. In *Advances in Neural Information Processing Systems*, 2025b. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/0418973e545b932939302cb605d06f43-Abstract-Conference.html.

Jinbang Huang, Yixin Xiao, Zhanguang Zhang, Mark Coates, Jianye Hao, and Yingxue Zhang. One Demo Is All It Takes: Planning Domain Derivation with LLMs from A Single Demonstration. In *International Conference on Learning Representations*, volume 2026, pp. 2489–2530, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/052483e6d5097f8d1892f04bfc90da41-Abstract-Conference.html.

Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. ReKep: Spatio-Temporal Reasoning of Relational Keypoint Constraints for Robotic Manipulation. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 4573–4602. PMLR, 2025c. URL https://proceedings.mlr.press/v270/huang25g.html.

Yuheng Ji, Huajie Tan, Jiayu Shi, Xiaoshuai Hao, Yuan Zhang, Hengyuan Zhang, Pengwei Wang, Mengdi Zhao, Yao Mu, Pengju An, Xinda Xue, Qinghang Su, Huaihai Lyu, Xiaolong Zheng, Jiaming Liu, Zhongyuan Wang, and Shanghang Zhang. RoboBrain: A Unified Brain Model for Robotic Manipulation from Abstract to Concrete. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1724–1734, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Ji_RoboBrain_A_Unified_Brain_Model_for_Robotic_Manipulation_from_Abstract_CVPR_2025_paper.html.

Yuanchen Ju, Yongyuan Liang, Yen-Jen Wang, Gireesh Nandiraju, Yuanliang Ju, Seungjae Lee, Qiao Gu, Elvis Hsieh, Furong Huang, and Koushil Sreenath. MomaGraph: State-aware unified scene graphs with vision-language models for embodied task planning. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://iclr.cc/virtual/2026/poster/10011623.

Amita Kamath, Cheng-Yu Hsieh, Kai-Wei Chang, and Ranjay Krishna. The hard positive truth about vision-language compositionality. In *Computer Vision – ECCV 2024*, 2024. URL https://www.ecva.net/papers/eccv_2024/papers_ECCV/papers/02149.pdf.

Hadi Khalaf, Claudio Mayrink Verdun, Alex Oesterling, Himabindu Lakkaraju, and Flavio Calmon. Inference-Time Reward Hacking in Large Language Models. In *Advances in Neural Information Processing Systems*, volume 38, pp. 61720–61760. Curran Associates, Inc., 2025. doi: 10.52202/085713-2064. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/590a0cc0306c1c63e2d66a51a407718f-Abstract-Conference.html.

Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen, Tony Lee, Li Li, Ruohan Zhang, Weiyu Liu, Percy Liang, Li Fei-Fei, Jiayuan Mao, and Jiajun Wu. Embodied Agent Interface: Benchmarking LLMs for Embodied Decision Making. In *Advances in Neural Information Processing Systems*, volume 37, pp. 100428–100534. Curran Associates, Inc., 2024. doi: 10.52202/079017-3188. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/b631da756d1573c24c9ba9c702fde5a9-Abstract.html.

Tianle Li, Jihai Zhang, Yongming Rao, and Yu Cheng. Unveiling the compositional ability gap in vision-language reasoning model. In *Advances in Neural Information Processing Systems*, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/b81d5608b42518f8f5b81a71a59509aa-Abstract-Conference.html.

Yichao Liang, Nishanth Kumar, Hao Tang, Adrian Weller, Joshua B Tenenbaum, Tom Silver, Joao F. Henriques, and Kevin Ellis. VisualPredicator: Learning Abstract World Models with Neuro-Symbolic Predicates for Robot Planning. In *International Conference on Learning Representations*, volume 2025, pp. 60416–60444, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/975db59bfa6eba2175a410f6afcecd99-Abstract-Conference.html.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison 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*, pp. 39578–39601, 2024. URL https://proceedings.iclr.cc/paper/2024/hash/aca97732e30bcf1303bc22ac3924fd16-Abstract-Conference.html.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Computer Vision – ECCV 2014*, volume 8693 of *Lecture Notes in Computer Science*, pp. 740–755. Springer, 2014. doi: 10.1007/978-3-319-10602-1_48. URL https://link.springer.com/chapter/10.1007/978-3-319-10602-1_48.

Ziyu Liu, Zeyi Sun, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Haodong Duan, Dahua Lin, and Jiaqi Wang. Visual-RFT: Visual Reinforcement Fine-Tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2034–2044, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Liu_Visual-RFT_Visual_Reinforcement_Fine-Tuning_ICCV_2025_paper.html.

Fan Lu, Wei Wu, Kecheng Zheng, Shuailei Ma, Biao Gong, Jiawei Liu, Wei Zhai, Yang Cao, Yujun Shen, and Zheng-Jun Zha. Benchmarking large vision-language models via directed scene graph for comprehensive image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19618–19627, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Lu_Benchmarking_Large_Vision-Language_Models_via_Directed_Scene_Graph_for_Comprehensive_CVPR_2025_paper.html.

Ruilin Luo, Zhuofan Zheng, Lei Wang, Yifan Wang, Xinzhe Ni, Zicheng Lin, Songtao Jiang, Yiyao Yu, Chufan Shi, Ruihang Chu, Jin Zeng, and Yujiu Yang. Unlocking Multimodal Mathematical Reasoning via Process Reward Model. In *Advances in Neural Information Processing Systems*, volume 38, pp. 49851–49899. Curran Associates, Inc., 2025. doi: 10.52202/085713-1664. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/4757e472d97ca980c24a487622f7ff00-Abstract-Conference.html.

Meta. Llama 3.2 Vision Model Card, 2024. URL https://huggingface.co/meta-llama/Llama-3.2-11B-Vision-Instruct. Released September 25, 2024.

Yingqian Min, Kun Zhou, Yifan Li, Yuhuan Wu, Han Peng, Yifan Du, Wayne Xin Zhao, Min Yang, and Ji-Rong Wen. Improving Vision-language Models with Perception-centric Process Reward Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 33099–33109, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Min_Improving_Vision-language_Models_with_Perception-centric_Process_Reward_Models_CVPR_2026_paper.html.

Ruaridh Mon-Williams, Gen Li, Ran Long, Wenqian Du, and Christopher G. Lucas. Embodied large language models enable robots to complete complex tasks in unpredictable environments. *Nature Machine Intelligence*, 7(4):592–601, 2025. doi: 10.1038/s42256-025-01005-x. URL https://www.nature.com/articles/s42256-025-01005-x.

Yao Mu, Junting Chen, Qing-Long Zhang, Shoufa Chen, Qiaojun Yu, Chongjian Ge, Runjian Chen, Zhixuan Liang, Mengkang Hu, Chaofan Tao, Peize Sun, Haibao Yu, Chao Yang, Wenqi Shao, Wenhai Wang, Jifeng Dai, Yu Qiao, Mingyu Ding, and Ping Luo. RoboCodeX: Multimodal Code Generation for Robotic Behavior Synthesis. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 36434–36454. PMLR, 2024. URL https://proceedings.mlr.press/v235/mu24a.html.

Fei Ni, Jianye Hao, Shiguang Wu, Longxin Kou, Yifu Yuan, Zibin Dong, Jinyi Liu, Mingzhi Li, Yuzheng Zhuang, and Yan Zheng. PERIA: Perceive, Reason, Imagine, Act via Holistic Language and Vision Planning for Manipulation. In *Advances in Neural Information Processing Systems*, volume 37, pp. 17541–17571. Curran Associates, Inc., 2024. doi: 10.52202/079017-0558. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/1f6af963e891e7efa229c24a1607fa7f-Abstract-Conference.html.

Minheng Ni, Zhengyuan Yang, Linjie Li, Chung-Ching Lin, Kevin Lin, Wangmeng Zuo, and Lijuan Wang. Point-RFT: Improving Multimodal Reasoning with Visually Grounded Reinforcement Finetuning. In *Advances in Neural Information Processing Systems*, volume 38, pp. 20538–20559. Curran Associates, Inc., 2025. doi: 10.52202/085713-0692. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/1d83ad88759cef8192451543e5d59bf6-Abstract-Conference.html.

Vignesh Ramanathan, Anmol Kalia, Vladan Petrovic, Yi Wen, Baixue Zheng, Baishan Guo, Rui Wang, Aaron Marquez, Rama Kovvuri, Abhishek Kadian, Amir Mousavi, Yiwen Song, Abhimanyu Dubey, and Dhruv Mahajan. PACO: Parts and attributes of common objects. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7141–7151, 2023. URL https://openaccess.thecvf.com/content/CVPR2023/html/Ramanathan_PACO_Parts_and_Attributes_of_Common_Objects_CVPR_2023_paper.html.

Darshana Saravanan, Varun Gupta, Darshan Singh, Zeeshan Khan, Vineet Gandhi, and Makarand Tapaswi. VELOCITI: Benchmarking video-language compositional reasoning with strict entailment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18914–18924, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Saravanan_VELOCITI_Benchmarking_Video-Language_Compositional_Reasoning_with_Strict_Entailment_CVPR_2025_paper.html.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models, 2024. URL https://arxiv.org/abs/2402.03300.

Chan Hee Song, Valts Blukis, Jonathan Tremblay, Stephen Tyree, Yu Su, and Stan Birchfield. RoboSpatial: Teaching Spatial Understanding to 2D and 3D Vision-Language Models for Robotics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15768–15780, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Song_RoboSpatial_Teaching_Spatial_Understanding_to_2D_and_3D_Vision-Language_Models_CVPR_2025_paper.html.

Huajie Tan, Yuheng Ji, Xiaoshuai Hao, Xiansheng Chen, Pengwei Wang, Zhongyuan Wang, and Shanghang Zhang. Reason-RFT: Reinforcement Fine-Tuning for Visual Reasoning of Vision Language Models. In *Advances in Neural Information Processing Systems*, volume 38, pp. 5772–5822. Curran Associates, Inc., 2025. doi: 10.52202/085713-0204. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/08d70284b013c03ba89cd2b642bc864b-Abstract-Conference.html.

V Team, Wenyi Hong, Wenmeng Yu, Xiaotao Gu, Guo Wang, Guobing Gan, Haomiao Tang, Jiale Cheng, Ji Qi, Junhui Ji, et al. GLM-4.5V and GLM-4.1V-Thinking: Towards Versatile Multimodal Reasoning with Scalable Reinforcement Learning, 2025. URL https://arxiv.org/abs/2507.01006. arXiv:2507.01006.

Vijay Viswanathan, Yanchao Sun, Shuang Ma, Xiang Kong, Meng Cao, Graham Neubig, and Tongshuang Wu. Checklists Are Better Than Reward Models For Aligning Language Models. In *Advances in Neural Information Processing Systems*, volume 38, pp. 114728–114754. Curran Associates, Inc., 2025. doi: 10.52202/085713-3824. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/a6837c1dd021f76f1b4098e3722052a8-Abstract-Conference.html.

Zifu Wan, Yaqi Xie, Ce Zhang, Zhiqiu Lin, Zihan Wang, Simon Stepputtis, Deva Ramanan, and Katia P. Sycara. InstructPart: Task-Oriented Part Segmentation with Instruction Reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 24202–24227. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.1179. URL https://aclanthology.org/2025.acl-long.1179/.

Di Wu, Jiaxin Fan, Chloe Gu, Guanbo Wang, Wei Yin, Wenhao Li, and Bo Jin. On-policy Reinforcement Fine-tuning with Offline reward for Multi-step Embodied Planning. In *Proceedings*

of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 39277–39307. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.1822. URL https://aclanthology.org/2026.acl-long.1822/.

Zhutian Yang, Caelan Garrett, Dieter Fox, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Guiding Long-Horizon Task and Motion Planning with Vision Language Models. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 16847–16853. IEEE, 2025. doi: 10.1109/icra55743.2025.11128705. URL https://doi.org/10.1109/ICRA55743.2025.11128705.

Chunlin Yu, Hanqing Wang, Ye Shi, Haoyang Luo, Sibei Yang, Jingyi Yu, and Jingya Wang. SeqAfford: Sequential 3D Affordance Reasoning via Multimodal Large Language Model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1691–1701, 2025a. URL https://openaccess.thecvf.com/content/CVPR2025/html/Yu_SeqAfford_Sequential_3D_Affordance_Reasoning_via_Multimodal_Large_Language_Model_CVPR_2025_paper.html.

Tianyu Yu, Zefan Wang, Chongyi Wang, Fuwei Huang, Wenshuo Ma, Zhihui He, Tianchi Cai, Weize Chen, Yuxiang Huang, Yuanqian Zhao, et al. MiniCPM-V 4.5: Cooking Efficient MLLMs via Architecture, Data, and Training Recipe, 2025b. URL https://arxiv.org/abs/2509.18154. arXiv:2509.18154.

Wentao Yuan, Jiafei Duan, Valts Blukis, Wilbert Pumacay, Ranjay Krishna, Adithyavairavan Murali, Arsalan Mousavian, and Dieter Fox. RoboPoint: A Vision-Language Model for Spatial Affordance Prediction in Robotics. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 4005–4020. PMLR, 2025. URL https://proceedings.mlr.press/v270/yuan25c.html.

Yifu Yuan, Yaoting Huang, Xianze Yao, Yutong Li, Shuoheng Zhang, Linqi Han, Pengyi Li, Jiangeng Sun, Wenting Jia, Zhao Zhang, et al. Embodied-R1.5: Evolving Physical Intelligence via Embodied Foundation Models, 2026. URL https://arxiv.org/abs/2606.11324. arXiv:2606.11324.

Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic Control via Embodied Chain-of-Thought Reasoning. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 3157–3181. PMLR, 2025. URL https://proceedings.mlr.press/v270/zawalski25a.html.

Yinuo Zhao, Jiale Yuan, Zhiyuan Xu, Xiaoshuai Hao, Xinyi Zhang, Kun Wu, Zhengping Che, Chi Harold Liu, and Jian Tang. Training-free generation of temporally consistent rewards from vlms. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8133–8143, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Zhao_Training-free_Generation_of_Temporally_Consistent_Rewards_from_VLMs_ICCV_2025_paper.html.

Enshen Zhou, Jingkun An, Cheng Chi, Yi Han, Shanyu Rong, Chi Zhang, Pengwei Wang, Zhongyuan Wang, Tiejun Huang, Lu Sheng, and Shanghang Zhang. RoboRefer: Towards Spatial Referring with Reasoning in Vision-Language Models for Robotics. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 28404–28481. Curran Associates, Inc., 2025. doi: 10.52202/085713-0957. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/29416b66c2149872b9d1415a3fd2c5e0-Abstract-Conference.html.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, et al. InternVL3: Exploring Advanced Training and Test-Time Recipes for Open-Source Multimodal Models, 2025. URL https://arxiv.org/abs/2504.10479. arXiv:2504.10479.

APPENDIX

The appendices follow the evidence chain of the paper. Appendix A defines what a connected task program contains and how the records are constructed. Appendices B–C show how omissions, additions, and execution are scored. Appendix D fixes the comparisons; Appendices E–F ask what supervision and rescoring reveal about task coverage. Appendices G–H then connect the complete reward to the programs learned and show one intervention end to end.

A PROGRAM AND DATASET SPECIFICATION

A.1 PROGRAM SCHEMA AND ACCEPTED EXPRESSIONS

A tool must remain the same entity when it is selected, grasped, and moved. CAP-7-PD encodes these links through shared identifiers across objects, steps, states, bindings, and motions (Table 5). This lets the verifier check whether the action, its prerequisite, and its trajectory refer to the same task, rather than accepting each field independently.

Table 5: **Program schema.** Types and references for the seven fields of Eq. 1, followed by record-level metadata. Literal values `-`, `null` and `init` respectively denote no moving entity, no part, and an initial provider.

Field	Record structure	Meaning and constraints
`objects` ($\mathcal{O}$)	List of `[entity, role]`	An entity may have several roles, so entries need not denote distinct entities. The selected patient and annotated alternatives are explicit.
`steps` ($\mathcal{S}$)	List of `[id, operator, moving entity or "-"]`	*IDs:* `task_NN` for main steps; `subtask_NN` for members of a `REPEAT_UNTIL` controller. *Controllers:* no states or moving entity; excluded from L. Members are executable.
`deps` (E)	List of `[before, after]` step IDs	**Required precedence.** Acyclic; every step except the first has an incoming edge. No path between two steps means they are unordered.
`lin` (L)	List of executable step IDs	**One execution order**: a linear extension of E. Includes controller members, excludes controllers. Earlier placement alone does not make a step a required predecessor.
`bindings` ($\mathcal{B}$)	List of `[step, affordance, entity, part or null, role, region]`	Multiple bindings may attach to one step or entity. Parts and regions name masks; e.g., the same glass can supply both a stabilization region and stir contact.
`states` (Σ)	Step ID $\rightarrow$ `[pre, add, del]`; each precondition is `[literal, provider]`	*Provider:* `init` or a step adding the literal; step providers precede consumers in L with no intervening deletion. *Literals:* entity, region and agent IDs; `!` negates, `=v` assigns a value. Controllers have no entry.
`traj` ($\mathcal{T}$)	Moving step $\rightarrow$ `[entity, rows]`; row = `[phase, x, y, theta, CH, progress]`	*Coordinates:* x, y in thousandths of image width/height; θ in degrees. *Progress:* non-decreasing $0 \rightarrow 1$; 8–12 rows per reference plan. *Contact:* X clear, N near, C contact, G grasped. *Height:* S support, T target, N near surface, E elevated, U/D transitioning.
Accept set	Selected patient, accepted instances, alternative/candidate/non-selected IDs	Only the object head credits accepted alternative instances in the selected-patient role. It does not rename bindings, state literals or trajectory subjects.
Record metadata	Split, family, query, goal, source ID, and schema version	IDs link the image, program, and training target so a record can be recovered without changing its content.

The accept set defines valid alternatives for the selected-patient object match. The other heads retain their requirements for step, binding, state, and trajectory identifiers. The schema also keeps two questions separate: E says which steps must precede others, while replay asks whether a chosen order can respect the declared resources (Appendices B.1 and C.1).

A.2 JOINT ANNOTATION, CONSTRUCTION, AND SPLITS

CAP-7-PD annotates photographs from COCO 2017 (Lin et al., 2014), PACO-LVIS (Ramanathan et al., 2023), and InstructPart (Wan et al., 2025). A proposal model writes each linked program; deterministic structural checks and a critic model examine its consistency. Acceptance requires agreement across the linked objects, steps, dependencies, states, and motions. Source and licensing metadata accompany each record; source images are not redistributed.

Table 6: **Dataset composition and splits.** (a) Released records and disjoint train–calibration–test partitions; the two indented rows are subsets of test. (b) Task-family counts in each split.

(a) Released records and evaluation subsets

Stage / split	**Count**	**Unit and use**
Released records	**1,698**	Records admitted after structural and semantic checks
Training	1,254	Learning targets; 14,833 linearized steps
Calibration	84	Validation and checkpoint selection
Test	**360**	Final evaluation and design analyses (Appendix G.3)
Unseen patient class	103	Test records whose patient class is absent from training
Unordered step pair	261	Test references to which the order-corruption family applies

(b) Task families by split

Task family	**Train**	**Calib.**	**Test**	**All**
serve_arrange	295	17	83	395
tidy_sort	214	13	61	288
cut_slice	216	8	60	284
place_stack	170	12	49	231
stir_scoop	164	15	48	227
pour	67	6	20	93
clean_wipe	60	3	17	80
spread	53	9	17	79
other	15	1	5	21
All	**1,254**	**84**	**360**	**1,698**

Image- and scene-disjoint splits prevent views of the same scene from appearing on both sides of the training–test boundary. Every split includes the full program annotation, allowing the sequence and program targets to be compared on the same tasks.

A.3 A COMPLETE ANNOTATED EXAMPLE

The stirring task in Figure 9 illustrates why the fields must be linked. The grasp step establishes possession of the plastic fork; the stirring step names that step as its state provider and binds the fork to its motion. Recognizing the fork and naming a stirring action would leave this prerequisite unchecked. The dependency and state fields make the connection explicit.



Figure 9: **One jointly annotated record.** (a) Photograph, entity masks (dashed: accepted alternative), instruction, objects and accept set; (b) steps; (c) required dependencies and committed order; (d) states of `task_04` and `task_07`; (e) bindings of `task_07`. Colours and step chips link panels; the dashed controller is absent from the committed order. Yellow traces `grasping(agent_01,plastic_fork_01)`, added by `task_04` and required by `task_07` from that provider.

B Complete Verification and Theoretical Details

Two choices determine whether an omitted requirement is visible: which obligations enter the score, and which alternatives the prediction must be distinguished from. Complete heads retain missing obligations and count unsupported additions. Deletion contrasts then test whether the prediction describes the intact task or a task with requirements removed. These choices address different parts of verification.

B.1 What counts as a task requirement?

The annotated reference $P^\star$ defines the task requirements to be checked. Objects match by (entity, role) within the accept set, bindings by typed tuple, and steps by a shared maximum-weight one-to-one assignment μ under Eq. (6); only positive-score pairs enter. Physical reference steps $\Phi^\star$ exclude controllers and follow $L^\star$; $C^\star$ is the closure of $E^\star$ over all nodes, controllers included, and $U^\star$ contains all unordered two-step pairs of $L^\star$. Table 7 summarizes the heads.

Table 7: **Accounting in the seven complete heads.** Reference obligations determine recall; generated content determines precision. Matching and per-element scores are defined below. The active heads are fixed by the reference contract; all seven are active on every test item.

Head	Reference obligations	Generated-side count	Empty set / matching rule	Eq.
q_O	Every (entity, role), with accepted alternatives for the selected patient	Every object entry	No reference entry: 1; partial matches contribute their score	(5)
q_B	Every typed binding	Every binding	No reference binding: 1; unresolved region: 0; shared step match and entity/role/affordance/part gate	(6)
q_S	Every step, including controllers; admissible position interval	Every step	Operator mismatch: no pair; shared assignment μ reused by all heads	(6)
q_E	Direct reference edges for SAT; **all reference pairs** for ORD3	**Every extra pair** with an endpoint not matched to a physical reference step	No must-precede edge: SAT $= 1$; zero ORD3 denominator: 1; unmatched endpoint: pair credit 0	(7)
q_C	Preconditions and effects of **all physical reference steps**	**All written** preconditions, additions and deletions	No preconditions on either side: support 1; no effects on either side: effect 1. Literals on unmatched/controller steps count as written only	(8)
q_T	**Every reference trajectory**	**Extra trajectories** not matched to a reference-trajectory step	No reference trajectory: 1; missing/wrong moving entity: 0	(9)
q_F	**Every reference trajectory** under fixed geometry and masks	Same as q_T	Same as q_T; frame, avoidance, goal and contact checks are conjunctive	(9)

$$F_{\text{soft}}(A,G) = \frac{2\sum_{(i,j)\in\pi} s_{ij}}{|A|+|G|} \quad (=1 \text{ if } |A|=|G|=0), \qquad \pi = \arg\max_{\text{one-to-one}} \sum_{(i,j)} s_{ij}, \tag{5}$$

where A, G are predicted/reference elements, π keeps positive-score pairs, and $1[\cdot]$ is the indicator. For non-empty references, q_O, q_B, q_S use F_{soft} (empty-reference overrides: Table 7) with

$$\begin{aligned} s^O_{ij} &= 1[r_i = r_j]\; 1[e_i \in \text{acc}(e_j, r_j)], \\ s^B_{ij} &= 1[\text{gate}_{ij}]\big(0.8\,\text{IoU}_{ij} + 0.2\,e^{-(d_{ij}/\sigma_c)^2}\big), \qquad \sigma_c = 0.10, \\ s^S_{ij} &= 1[\text{op}_i = \text{op}_j]\big(1 - \tfrac{1}{2}\,1[\text{mov}_i \neq \text{mov}_j]\big)\big(1 - \tfrac{1}{2}\,\Delta_{ij}\big). \end{aligned} \tag{6}$$

Here $\text{acc}(e_j, r_j) = \{e_j\}$ except when r_j is `selected_patient`, for which it is the annotated accept set; gate_{ij} requires $\mu(\text{step}_i) = \text{step}_j$ and equal affordance, entity, role, and part when the reference names one; d_{ij} is the centroid distance over the image diagonal; the moving-entity factor applies only when the reference step names a moving entity; Δ_{ij} is the distance from the normalised position $\text{pos}(i)/\max(|\hat{L}|-1, 1)$ to the interval $[\,|\text{anc}(j)|,\; n-1-|\text{desc}(j)|\,]/(n-1)$ of positions reference step j occupies over all linear extensions of $E^\star$ ($n = |L^\star|$; use denominator 1 for $n = 1$; control nodes span $[0, 1]$). Dependency-consistent alternatives have $\Delta_{ij} = 0$; SAT additionally

checks violations that position intervals can miss.

$$\begin{aligned}
\mathrm{SAT} &= \frac{1}{|E^\star_{\mathrm{mp}}|} \sum_{(a,b)\in E^\star_{\mathrm{mp}}} 1\big[\mu^{-1}(a), \mu^{-1}(b) \in \hat{L},\ \mathrm{pos}(\mu^{-1}a) < \mathrm{pos}(\mu^{-1}b)\big], \\
N_{\mathrm{correct}} &= \sum_{\{a,b\}\in U^\star} 1\big[\mu^{-1}(a), \mu^{-1}(b) \in \hat{L},\ \mathrm{cls}_{\hat{C}}(\mu^{-1}a, \mu^{-1}b) = \mathrm{cls}_{C^\star}(a,b)\big], \\
N_{\mathrm{extra}} &= um + \tbinom{u}{2}, \qquad \mathrm{ORD3} = \frac{N_{\mathrm{correct}}}{|U^\star| + N_{\mathrm{extra}}}, \qquad q_E = H_\varepsilon(\mathrm{SAT}, \mathrm{ORD3}), \\
H_\varepsilon(x_1, \ldots, x_k) &= \frac{k}{\sum_{j=1}^k (x_j + \varepsilon)^{-1}} - \varepsilon, \qquad \varepsilon = 0.02,
\end{aligned} \tag{7}$$

where $E^\star_{\mathrm{mp}}$ are the direct $E^\star$ edges with both endpoints in $L^\star$ ($\mathrm{SAT} = 1$ when there is none), $\mathrm{cls} \in \{a < b,\ b < a,\ \text{incomparable}\}$ is read from the respective closure ($\hat{C}$: closure of $\hat{E}$ over all predicted steps), m is the number of steps of $\hat{L}$ matched to $L^\star$ steps and $u = |\hat{L}| - m$ the number of remaining steps (padding, duplicates, foreign operators, steps matched to control nodes); $\mathrm{ORD3} = 1$ when its denominator is 0. Pairs with unmatched endpoints or endpoints outside $\hat{L}$ remain in $U^\star$ with credit 0; missing explicit edges do not remove closure-classified pairs.

$$\begin{aligned}
P_{\mathrm{pre}} &= \frac{\#\{(s,p):\ \mu(s) \in \Phi^\star,\ p \in \mathrm{pre}(\mu(s))\}}{\#\{(s,p)\ \text{written}\}}, \\
R_{\mathrm{pre}} &= \frac{\#\{(s,p):\ \mu(s) \in \Phi^\star,\ p \in \mathrm{pre}(\mu(s)),\ \text{provider valid}\}}{\sum_{g\in\Phi^\star} |\mathrm{pre}(g)|}, \\
\mathrm{support} &= \frac{2P_{\mathrm{pre}}R_{\mathrm{pre}}}{P_{\mathrm{pre}} + R_{\mathrm{pre}}}, \qquad \mathrm{effect} = \frac{2\,\mathrm{TP}_{\mathrm{eff}}}{\hat{n}_{\mathrm{eff}} + n^\star_{\mathrm{eff}}}, \qquad q_C = H_\varepsilon(\mathrm{support},\ \mathrm{effect}),
\end{aligned} \tag{8}$$

with $\mathrm{TP}_{\mathrm{eff}} = \sum_{s:\,\mu(s)\in\Phi^\star} \big(|\mathrm{add}(s) \cap \mathrm{add}(\mu s)| + |\mathrm{del}(s) \cap \mathrm{del}(\mu s)|\big)$, $\hat{n}_{\mathrm{eff}} = \sum_s (|\mathrm{add}(s)| + |\mathrm{del}(s)|)$ over all predicted steps and $n^\star_{\mathrm{eff}} = \sum_{g\in\Phi^\star} (|\mathrm{add}(g)| + |\mathrm{del}(g)|)$. Written pairs (s,p) include all predicted states: unmatched/controller-step and non-reference preconditions lower precision only. Recall counts all physical reference preconditions, including unmatched steps. When the reference has preconditions but the prediction writes none, $P_{\mathrm{pre}} = R_{\mathrm{pre}} = 0$. Support is 1 when neither side has a precondition and 0 when $P_{\mathrm{pre}} + R_{\mathrm{pre}} = 0$; effect is 1 when neither side has effects. A provider u named for p on step s is valid when u is the initial state, p belongs to the reference initial fact set $\sigma^\star_0$, and no step before s in $\hat{L}$ deletes p; or when u is a step that is an ancestor of s in the closure of $\hat{E}$, precedes s in $\hat{L}$, adds p, and no step strictly between them in $\hat{L}$ deletes p.

$$q_T = \frac{\sum_{t\in\mathcal{T}^\star} s_t}{|\mathcal{T}^\star| + N^{\mathrm{extra}}_{\mathcal{T}}}, \qquad q_F = \frac{\sum_{t\in\mathcal{T}^\star} f_t}{|\mathcal{T}^\star| + N^{\mathrm{extra}}_{\mathcal{T}}}, \qquad q_T = q_F = 1 \text{ if } \mathcal{T}^\star = \emptyset, \tag{9}$$

where $s_t = 0.35\,\mathrm{position} + 0.15\,\mathrm{angle} + 0.20\,\mathrm{phase} + 0.15\,\mathrm{contact} + 0.15\,\mathrm{height}$ scores the predicted trajectory on the step matched to t when it moves the same entity and is 0 otherwise, $f_t = \mathrm{frame} \cdot \mathrm{avoid} \cdot \mathrm{goal} \cdot \mathrm{contact} \in \{0,1\}$ is the sweep-contract check of that trajectory, and $N^{\mathrm{extra}}_{\mathcal{T}}$ counts predicted trajectories on steps not matched to a step that carries a reference trajectory. With no reference trajectory both heads return 1, regardless of predictions.

The reference fixes active heads $\mathcal{H}(P^\star)$ (all seven on test); their mean is M_{cc} and the gate-free score $S = 2M_{\mathrm{cc}} - 1$. Training adds gates and risk discount (Appendix C). $N_{\mathrm{expressed}}$ counts $U^\star$ pairs with both steps validly matched into $\hat{L}$. Per-program and pooled-count factorizations hold; means of their factors need not multiply (Proposition B.4):

$$\mathrm{ORD3} = \frac{N_{\mathrm{expressed}}}{D_{\mathrm{ORD3}}} \cdot \frac{N_{\mathrm{correct}}}{N_{\mathrm{expressed}}}, \qquad D_{\mathrm{ORD3}} = |U^\star| + N_{\mathrm{extra}}, \qquad N_{\mathrm{expressed}} > 0. \tag{10}$$

ORD3 readout conventions. If $D_{\mathrm{ORD3}} = 0$, ORD3 is 1. With fewer than two validly matched reference steps, $N_{\mathrm{expressed}} = N_{\mathrm{correct}} = 0$: ORD3 and coverage are 0, the denominator remains in pooled counts, and only the undefined item-level conditional correctness is excluded. The factorization distinguishes two reasons for low order agreement: too little of the required relation set is expressed, or the expressed relations are classified incorrectly. Conditional correctness alone cannot distinguish an accurate fragment from an accurate full plan. We therefore pair it with coverage under the complete denominator and report reference-only step coverage separately (Tables 14 and 4).

B.2 WHY OMISSIONS AND ADDITIONS CANNOT HIDE

The propositions make the two shortcuts explicit. A program can leave a required relation unmatched, or add structure that the reference never asks for. The first two results show how complete accounting charges these moves; the next two explain why deletion contrasts and the coverage readout are needed to expose them. The excess-risk statement for the training reward appears in Appendix C.

Proposition B.1 (A missing step is charged by complete accounting). Fix the reference, the assignment on remaining steps and their support claims. Let $s^\star \in \Phi^\star$ have $d = |\mathrm{pre}(s^\star)| \geq 1$ and $\mu^{-1}(s^\star) = \emptyset$. (a) The original support term equals N/D, where D counts the reference preconditions of the matched physical steps and N those among them written with a valid provider; $\mathrm{pre}(s^\star)$ enters neither count, so the term equals 1 whenever every remaining obligation is met. If, moreover, $\hat{P}$ arises from a prediction $\hat{P}_0$ by deleting the step $\hat{s} = \mu_0^{-1}(s^\star)$, and the assignment and the validity verdicts of the remaining claims are unchanged, then the term moves from N_0/D_0 to $(N_0 - a)/(D_0 - d)$, with a the number of valid claims on $\hat{s}$; when $D_0 > d$, the new value is at least the old one iff $a/d \leq N_0/D_0$. If deletion leaves no matched obligation, the implementation uses its empty-scope convention instead of this ratio. (b) In Eq. (8), $R_{\mathrm{pre}} \leq 1 - d/\sum_{g \in \Phi^\star} |\mathrm{pre}(g)| < 1$; hence $\mathrm{support} < 1$ and $q_C < 1$. (c) If $s^\star \in L^\star$ and $|L^\star| \geq 2$, the $|L^\star| - 1$ pairs of $U^\star$ containing $s^\star$ score 0, so $\mathrm{ORD3} \leq 1 - (|L^\star| - 1)/D_{\mathrm{ORD3}} < 1$ and $q_E < 1$.

Proof. (a) Original support sums only matched physical steps, excluding the d obligations of $s^\star$. Deletion removes d from its denominator and a valid claims from its numerator. For $D_0 > d$, cross-multiplication gives $(N_0 - a)/(D_0 - d) \geq N_0/D_0 \iff N_0 d \geq a D_0$. The assumed verdict invariance holds, for example, when no remaining provider path traverses $\hat{s}$ and it deletes no subsequently used literal. (b) No prediction can credit $s^\star$'s d obligations without matching it, giving the recall bound. For $0 < P \leq 1$ and $R < 1$, $2PR < P + R$; for $P = 0$, support is 0. Strict monotonicity of H_ε, with $H_\varepsilon(1, 1) = 1$, gives $q_C < 1$. (c) All $|L^\star| - 1$ pairs containing $s^\star$ fail the both-matched test, giving the stated ORD3 bound and $q_E < 1$. □

Proposition B.2 (Unsupported additions cannot improve complete accounting). Let $\hat{P}'$ be $\hat{P}$ plus one element e of one of the following kinds: (i) an object entry or a binding with no positive-score pair; (ii) a step in $\hat{L}$ that is not matched to an $L^\star$ step, inserted so that the relative order of the matched steps and the closure $\hat{C}$ restricted to matched steps are unchanged; (iii) a precondition literal written on an unmatched step, on a step matched to a control node, or outside $\mathrm{pre}(\mu(s))$ of its step; (iv) an add or delete literal outside the corresponding reference effects of its matched step; (v) a trajectory on a step not matched to a reference-trajectory step; (vi) a dependency edge whose only change to $\hat{C}$ on matched steps is to make reference-incomparable pairs comparable. Assume that the assignment μ, the existing positive match weights, and all support/effect success verdicts not directly targeted by the edit are unchanged. In case (ii), the added step has no positive-score reference match; in cases (i) and (v), the reference set for the affected head is non-empty. Then each affected count ratio a/d, with $a \geq 0$ and $d > 0$, becomes $a/(d + k)$ with $k \geq 1$ in cases (i)–(v), or $(a - k)/d$ with $k \geq 0$ in case (vi). For case (ii), the order denominator grows by $k = |\hat{L}|$ before insertion when $\hat{L}$ is non-empty. Every affected head is non-increasing; heads whose inputs and success verdicts are unchanged stay fixed, so M_{cc} does not increase. A denominator ratio decreases strictly when its numerator is positive. For a precondition addition, strict decrease of the support term also requires $R_{\mathrm{pre}} > 0$; for an edge addition, it requires at least one previously correct incomparable pair to change class. Empty-reference conventions can instead leave a head at 1.

Proof. (i) The zero-score addition leaves $T = \sum_\pi s_{ij}$ fixed and increases $|A|$ in $F_{\mathrm{soft}} = 2T/(|A| + |G|)$ by 1. (ii) Preserved matched-step closure and order fix N_{correct} and SAT. Extra pairs increase by $(u + 1)m + \binom{u+1}{2} - um - \binom{u}{2} = m + u = |\hat{L}|$. Fixed match weights also fix q_S's numerator while its denominator grows. (iii) The written precondition count grows by 1 with both success counts fixed; hence P_{pre} falls or stays fixed, R_{pre} is fixed, and their F1 cannot increase. (iv) The effect denominator grows by 1 with fixed numerator. (v) $N_{\mathcal{T}}^{\mathrm{extra}}$ grows by 1 with fixed $\sum_t s_t, \sum_t f_t$. (vi) The k newly comparable reference-incomparable pairs lose their correct classifications; D_{ORD3} stays fixed. Verdict invariance excludes simultaneously repairing a provider path. Monotonicity of H_ε and the equal-weight mean proves the claim. With fixed gate and excess risk, reward is non-

decreasing in M_{cc}: its slope is non-negative if $1 - \beta e \geq 0$; otherwise clipping yields -1 throughout $M_{\mathrm{cc}} \geq 0$. □

The original accounting leaves several generated elements outside the denominator: extra preconditions, states on unmatched steps, extra trajectories, and pairs involving unmatched steps. The constructed padding tests evaluate these full-program effects in Appendix B.4.

Proposition B.3 (Registered corruptions cannot expose an omitted dependency). Fix $\hat{P}$ to the reference steps, bindings, states, trajectories and order, with $\hat{E} \subseteq E^\star$ and $\hat{L} = L^\star$. Assume identity step matching against both the reference and its applicable corruption γ. Then $q_h(\hat{P}; \gamma P^\star) \leq q_h(\hat{P}; P^\star)$ for every head h and every applicable γ of the role, inst, ord and mot families; hence $S(\hat{P}; \gamma P^\star) \leq S(\hat{P}; P^\star)$ and the outcome is a win or a tie.

Proof. Identity matching gives $q_O = q_B = q_S = q_T = q_F = 1$ against $P^\star$. *role* changes only binding labels, so only $q_B \leq 1$ changes. *inst* renames the selected instance in objects, bindings/regions, states, trajectories and moving entities and removes the original from the accept set. Object, binding, trajectory and moving-entity credits can only fall; q_C's denominators stay fixed while renamed literals cannot increase its success counts. The unchanged matching, dependencies and orders fix q_E. *ord* replaces physical-step edges by the chain of an alternative linear extension L', keeping $L^\star$. Since $\hat{C} \subseteq C^\star$, predicted classes are correct directions or incomparable on $C^\star$, and incomparable on reference-incomparable pairs. The chain preserves $C^\star$'s directions and makes every other pair comparable, so N_{correct} cannot rise; its denominator is fixed. Original $\mathrm{SAT} = 1$ cannot rise; position intervals shrink to L''s points, so Δ_{ij} cannot fall. Support does not read reference edges; other fields and their identity matches are unchanged. *mot* changes only trajectories, whose original credits are all 1. Thus each head and their monotone aggregate can only fall. □

A deletion corruption reverses the inclusion: after `omit_edge` the reference closure loses comparabilities, a prediction with $\hat{E} = \emptyset$ agrees with $\gamma P^\star$ on more pairs than with $P^\star$, and it loses whenever the margin exceeds 10^{-9} (Table 9, deletion columns).

Proposition B.4 (Order agreement separates coverage from conditional correctness). For every program, $N_{\mathrm{correct}} \leq N_{\mathrm{expressed}} \leq |U^\star| \leq D_{\mathrm{ORD3}}$, and for $N_{\mathrm{expressed}} > 0$ Eq. (10) holds with both factors in $[0, 1]$. For a set of items with $\sum_i N_{\mathrm{expressed},i} > 0$,

$$\frac{\sum_i N_{\mathrm{correct},i}}{\sum_i D_{\mathrm{ORD3},i}} = \frac{\sum_i N_{\mathrm{expressed},i}}{\sum_i D_{\mathrm{ORD3},i}} \cdot \frac{\sum_i N_{\mathrm{correct},i}}{\sum_i N_{\mathrm{expressed},i}}, \tag{11}$$

On a common item set with $N_{\mathrm{expressed},i} > 0$ for every item, the mean of ORD3 equals the product of the means of the two factors plus their empirical covariance. The coverage mean, conditional-correctness mean, and covariance in this identity all use that common item set.

Proof. Correct pairs are expressed, expressed pairs belong to $U^\star$, and $D_{\mathrm{ORD3}} = |U^\star| + N_{\mathrm{extra}}$. Multiplying and dividing by the expressed count gives both identities; $\overline{ca} = \bar{c}\bar{a} + \mathrm{Cov}(c, a)$ gives the common-item mean. □

Reference self-score. For reference self-scoring, identity matching, identical closure and trajectories, and zero extra pairs give every head 1 provided the recorded providers satisfy Eq. (8) along $L^\star$. The recorded test self-score confirms minimum head 1.000 on 360/360 references (Table 9); this calibrates condition (C3) for the reference-free terms in Appendix C.

B.3 How are the contrasts constructed?

To test whether a verifier recognizes a binding rather than a collection of correct fields, we edit one relation in the reference and keep its masks, initial facts, and active heads fixed. A prediction is then compared with the original and edited references. No-op edits are dropped, and a family mean uses only items for which that relation changes (Table 8).

Table 8: **Reference-derived corruptions and comparison protocols.** A fixed prediction is scored against the original and edited reference. Applicability requires the named relation to change; the lower panel defines reported family sets.

Family	Edited object (reference side)	Retained	Relation under test	Applicability (else no-op)	Head	Items
Registered modification families (four)						
role	two (affordance, role) label groups of one entity's bindings swapped	masks, parts, objects, steps, order, states, trajectories	which region carries which role	entity with ≥ 2 label groups	q_B	306
inst	selected patient $\rightarrow$ same-class instance outside the accept set; renamed in objects, bindings (region of that instance), state arguments, trajectory subject, moving entity; accept set re-centred	roles, steps, order, dependencies, other objects	which instance the task acts on	a same-class instance outside the accept set	q_O	86
ord	$E^\star \rightarrow$ total-order chain along a linear extension $L' \neq L^\star$ (variant 0 anti-$L^\star$, others random); controller edges kept; $L^\star$ unchanged	steps, order, bindings, states, trajectories	reference-incomparable pairs become comparable; $C^\star$ directions kept; must-precede set $=$ chain	$\lvert I^\star \rvert > 0$	q_E	261
mot	2pt: trajectory reduced to its endpoints; subject: moving entity swapped; dup: endpoint poses duplicated, row count kept	steps, order, bindings, states	trajectory shape or moving entity	a trajectory with > 2 rows; another entity (subject)	q_T	360
Other original families						
composite	inst+ord (even variants) or role+mot(2pt) (odd variants)	as its parts	two relations at once	both parts applicable	—	360
bind	bindings re-attached to another step (even) or another part (odd)	masks, steps, order, states	which step or part a binding belongs to	≥ 2 steps with bindings, or an alternative part	q_B	350
Deletion family (added; Section 4.2)						
omit_edge	$k \in \{1, 2, 3\}$ transitive-reduction edges between $L^\star$ steps deleted; $L^\star$ unchanged	steps, order, states, bindings, trajectories, other edges	the deleted precedences become incomparable	$\geq k$ reduction edges	q_E	360
omit_step	one non-initial physical step deleted with its states, bindings, trajectory and order entry; predecessors spliced to successors	remaining steps and their fields	the deleted step's obligations	≥ 2 physical steps	q_S	360
omit_state	pre, add and del of one physical step with literals emptied (entry kept)	steps, order, other states	the step's support and effects	a physical step with literals	q_C	360

Family sets and pair rules				
Set	**Families**	**Heads**	**Pair rule**	Reported in
held-out	inst, ord, composite	v2.2	τ + floor	Section 5 default
held-out, original (v1)	inst, ord, composite	v1	strict 10^{-9}	Figure 7a (Original score); Table 3
four	role, inst, ord, mot	v2.2	τ + floor	Table 3
four + deletion	role, inst, ord, mot, mean(omit_edge, omit_step, omit_state)	v2.2	τ + floor; strict for the deletion term	Table 3
deletion, step	omit_step	v2.2	strict 10^{-9}	Section 5.3
self-check	any	v1 / v2.2	strict 10^{-9}	verifier property, not a model readout

Items: test references (of 360) with at least one applicable, non-no-op variant of the family. Variants: three per original family and item before no-op removal; three for omit_edge ($k = 1, 2, 3$), one each for omit_step and omit_state (no no-op rows). The provider-rewrite family (pre) acts on the prediction side only and enters no reference set. Floor $\delta_\varepsilon = 0.0377$, $\tau = 0.0108$ on every test item (seven active heads). Unparsed predictions tie every pair. "Head" is the designated head of the floor rule; composite has none. Self-check: $S(\gamma P^\star; P^\star)$, the corrupted program scored as a prediction.

Each comparison fixes $\hat{P}_i$ and compares $S(\hat{P}_i; P_i^\star)$ with $S(\hat{P}_i; \gamma P_i^\star)$, S being the gate-free aggregate $2M_{\mathrm{cc}} - 1$ over the active heads. Modification families tie when the scores differ by at most τ of Eq. (12) or when the designated head is below the floor δ_ε on both sides; deletion families tie only within 10^{-9}, their margins being deterministic and often below τ; an unparsed prediction ties every pair. The separate self-check $S(\gamma P^\star; P^\star)$ scores a corrupted program as the prediction; it is a verifier property, outside the fixed-prediction CBA.

$$\begin{aligned} \tau &= \max\Big(\frac{2\,\delta_\varepsilon}{|\mathcal{H}(P^\star)|},\ \tau_{\mathrm{noise}}\Big), \\ \delta_\varepsilon &= H_\varepsilon(1,1,0) = \frac{3}{2/1.02 + 1/0.02} - 0.02 = 0.0377, \qquad \tau_{\mathrm{noise}} = 0.0078, \end{aligned} \tag{12}$$

so $\tau = 0.0108$ for seven active heads: δ_ε is the largest value the smoothed harmonic mean returns when one of three channels is exactly 0, mapped to the R scale through the equal-weight mean, and τ_{noise} is the largest absolute difference of the held-out readout between recorded same-model greedy re-decodes on the test split.

$$\mathrm{CBA}_{\mathcal{F}} = \frac{1}{|\mathcal{I}_{\mathcal{F}}|} \sum_{i \in \mathcal{I}_{\mathcal{F}}} \frac{1}{|\mathcal{F}_i|} \sum_{f \in \mathcal{F}_i} \frac{1}{|\Gamma_{if}|} \sum_{\gamma \in \Gamma_{if}} o_{i\gamma}, \qquad o_{i\gamma} \in \{0, \tfrac{1}{2}, 1\}, \tag{13}$$

where $\mathcal{F}_i$ are the families of the set applicable to item i, Γ_{if} their applicable variants, and $\mathcal{I}_{\mathcal{F}}$ the items with $\mathcal{F}_i \neq \emptyset$; in the four + deletion set the deletion component enters as one family-level term, the mean of its three families, so the set has five equal components.

B.4 Do deliberately incomplete programs still win?

The nine rewrites start from test references and introduce omission, padding, or over-ordering while keeping the underlying task fixed. The resulting comparisons test whether these edits exploit the scoring rule.

`deps_empty`, `single_edge` and `half_deps` keep every field and reduce $E^\star$ to no edge, its first edge between $L^\star$ steps, or every other edge; the chains replace $E^\star$ by the total order along $L^\star$ or along the ord family's L' (Table 8), keeping $\hat{L} = L^\star$; `pad_`k appends k foreign-operator steps without states, bindings or edges; `dup_3` repeats the first three physical steps with their states, each after its original and linked to it by one edge. Direct scores aggregate the corresponding scorer's active heads without execution gates; for complete heads, $S = 2M_{\mathrm{cc}} - 1$. Training adds the structural and execution gates in Appendix C. CBA columns follow Table 8.

Table 9 separates direct scores from contrast accuracy. A lower direct score need not make a prediction lose to the corrupted alternatives: the alternatives may still be less compatible with it than the intact reference.

The edgeless program exposes this distinction. Complete accounting lowers its order and state-support scores, yet its accuracy against modification-only contrasts remains near the reference. Those contrasts damage roles, instances, ordering, or motion without offering the missing-dependency alternative that would reveal its omission. An edge-deleted reference supplies that alternative: the edgeless prediction agrees with it more closely and loses the comparison. Thus, retaining obligations in the score and testing their deletion are complementary design choices.

The padding rows isolate a second mechanism. As unmatched steps accumulate, they create additional pairs without creating corresponding correct relations; order agreement falls even though most binding and motion scores stay high. This is the effect the complete denominator is designed to capture. The contrast score changes more slowly, particularly for the smallest padding edit, because it measures preference between references rather than absolute program completeness.

The chains reveal why submitted order and necessary dependencies must also remain separate. Both preserve the physical steps and satisfy the reference precedence in the submitted order, but they turn independent step pairs into ordered pairs. Pairwise agreement detects that loss of independence. The witness distinguishes the two chains further: ordering edges inconsistent with the submitted sequence can fail the submitted-order witness even when their aggregate relation scores match.

Table 9: **Complete scores and deletion contrasts expose different defects.** (A) The same nine rewrites are scored directly and compared against corrupted references. Darker cells show larger departures from the reference. (B) Adding deletion contrasts lowers every rewrite below its reference; points and intervals show the paired differences.

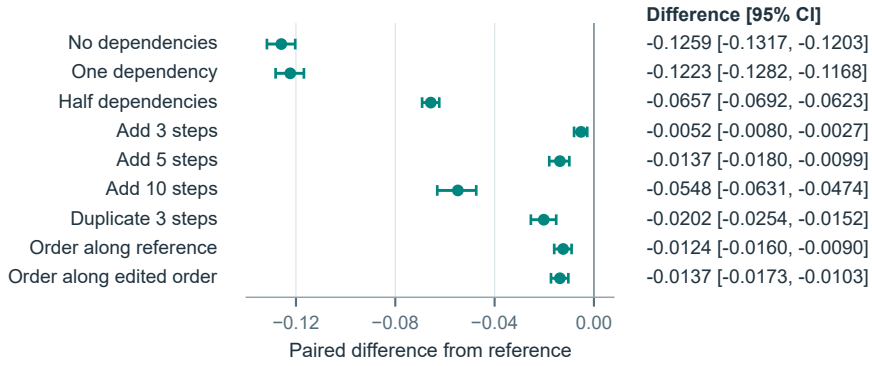
A Scoring and testing capture different defects

Lower direct scores need not imply lower contrast accuracy

Program	Direct original	Direct complete	Contrast original	Contrast complete	Four families	Four + deletion	Changed tasks
Reference	**1.0000**	**1.0000**	**1.0000**	**0.9990**	**0.9995**	**0.9997**	—
No dependencies	**0.8624**	**0.6761**	*1.0000*	*0.9958*	***0.9981***	**0.8738**	360
One dependency	0.8748	0.6969	*1.0000*	*0.9958*	*0.9981*	0.8773	360
Half dependencies	0.9594	0.8777	*1.0000*	*0.9958*	*0.9981*	0.9340	360
Add 3 steps	0.9477	0.8773	*0.9958*	*0.9973*	*0.9986*	0.9944 †	360
Add 5 steps	0.9160	0.8107	*0.9949*	0.9935	0.9943	0.9860	360
Add 10 steps	0.8313	0.6715	0.9799	0.9483	0.9742	0.9448	360
Duplicate 3 steps	0.9344	0.8399	0.9695	0.9878	0.9910	0.9795	360
Order along reference	0.7968	0.9754	0.5086	0.9642	0.9829	0.9873	261
Order along edited order	0.7968	0.9754	0.5086	0.9608	0.9812	0.9860	261

B Deletion contrasts lower all nine rewrite scores

Difference from the reference in the four-plus-deletion comparison



Means over 360 test items; changed-task counts are not scoring denominators. The reference row is each column's baseline. Original/complete denote the original and commitment-complete scorers. Contrast: instance, order, composite (original: strict margin; complete: τ + floor). Four: role, instance, order, motion (τ + floor). Four + deletion adds the mean of three deletion families under the strict margin (Table 8). Intervals: paired item-level bootstrap, $B = 5{,}000$. Italics: the paired interval against the reference includes 0. †: within the replicate-arm noise band (same-model SFT replicates: max $|\Delta|$ 0.004; interval half-width $\leq$ 0.012). Bold highlights the edgeless program's direct-score/contrast disagreement.

Table 9: **(continued) The checks identify what each rewrite changes.** (C) Complete relation readouts distinguish missing dependencies, unmatched steps, and lost independence. (D) Original checks, execution witnesses, and deletion tests separate errors that can look alike in an aggregate score. Darker score cells indicate larger decreases from the reference; counts have no color scale.

C Which relations does each rewrite affect?

Shading shows the decrease from the reference; labels give exact scores

	Steps	Bindings	SAT	ORD3	Order	Support	Trajectory	Sweep
Reference	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
No dependencies	1.000	1.000	**1.000**	**0.134**	0.210	**0.656**	1.000	1.000
One dependency	1.000	1.000	1.000	0.155	0.242	0.697	1.000	1.000
Half dependencies	1.000	1.000	1.000	0.501	0.651	0.921	1.000	1.000
Add 3 steps	0.845	0.998	0.995	0.600	0.748	0.995	0.994	0.992
Add 5 steps	0.767	0.990	0.982	0.454	0.620	0.979	0.990	0.992
Add 10 steps	0.628	0.959	0.930	0.254	0.401	0.922	0.966	0.976
Duplicate 3 steps	0.862	0.956	0.994	0.585	0.735	0.891	0.998	0.998
Order along reference	1.000	1.000	1.000	0.866	0.914	1.000	1.000	1.000
Order along edited order	1.000	1.000	1.000	0.866	0.914	1.000	1.000	1.000

D Original checks, added structure, and deletion tests

	Extra pairs	Original order	Original PRE	Witness passes	Delete edge	Delete step	Delete state
Reference	**0**	**1.000**	**1.000**	**360**	**1.000**	**1.000**	**1.000**
No dependencies	0	1.000	1.000	360	**0.000**	1.000	0.636
One dependency	0	1.000	1.000	360	0.025	1.000	0.650
Half dependencies	0	1.000	1.000	360	0.294	1.000	1.000
Add 3 steps	37.3	0.994	0.995	0	0.972	0.975	1.000
Add 5 steps	67.2	0.986	0.994	0	0.940	0.967	0.981
Add 10 steps	159.3	0.953	0.985	0	0.766	0.942	0.919
Duplicate 3 steps	37.3	0.973	0.962	217	0.900	0.975	0.972
Order along reference	0	0.289	0.275	360	1.000	1.000	1.000
Order along edited order	0	0.289	0.275	99	1.000	1.000	1.000

Counts are unshaded; other cells use one reference-relative scale. Witness: legal submitted order.

All means use the same 360 tasks. Steps, bindings, order, support, trajectory and sweep denote $q_S, q_B, q_E, q_C, q_T, q_F$; SAT and ORD3 are the two order channels. Extra pairs: mean N_{extra}. Original order/PRE: the original order head and its incomparability channel. Witness: number of submitted orders that replay under the program's declared states (Appendix C). Deletion entries are per-family means of win 1, tie $\frac{1}{2}$, and loss 0. Bold highlights the edgeless program's contrasting outcomes.

C Execution Semantics and Reward Implementation

A program can be consistent with the states it declares yet fail when executed from the reference scene. We therefore distinguish declared-state replay from execution under fixed operator rules and reference initial facts. Training uses the former; the latter tests whether the learned change carries beyond the program's own declarations. Task-goal achievement additionally checks the final outcome.

C.1 What does it mean for a program to execute?

The first replay asks whether a program is internally coherent. It uses the program's declared preconditions, effects, and `init` facts, follows its submitted order $\hat{L}$, records failures, and applies effects even after a failed step:

$$\sigma_0 = \{\, b \mapsto v : \ (b{=}v,\ \texttt{init}) \in \mathrm{pre}(s) \text{ for some step } s \,\},$$
$$\mathrm{ok}_t = \big[\sigma_{t-1}(b) = v \ \ \forall\, (b{=}v) \in \mathrm{pre}(\hat{L}_t) \setminus \mathrm{rc}\big] \wedge \mathrm{hand}(\hat{L}_t, \sigma_{t-1}), \tag{14}$$
$$\sigma_t = \big(\sigma_{t-1} \ominus \mathrm{del}(\hat{L}_t)\big) \oplus \mathrm{add}(\hat{L}_t), \qquad \mathrm{legal}(\hat{L}) = \textstyle\bigwedge_t \mathrm{ok}_t.$$

Facts map predicate bodies b to values v: bare literals mean true, `!p` false, and `p=v` value v. $\ominus$ removes a matching current value; $\oplus$ writes one. Preconditions exempt `region_clear` (rc). Hand

and transport rules reject actions that conflict with the declared held entity. Initial facts are self-declared; reference-provider validity is checked by q_C.

The submitted-order witness is

$$W(\hat{P}) = [\hat{L} \neq \emptyset] \wedge [\text{states cover } \hat{L}] \wedge \text{acyclic}(E|_{\hat{L}}) \wedge [\hat{L} \in \text{LE}(E|_{\hat{L}})] \wedge \text{legal}(\hat{L}), \quad (15)$$

Here $E|_{\hat{L}}$ restricts edges to listed steps and LE denotes linear extensions. W evaluates the submitted order, while $\widehat{\ell}$ measures the failure rate across sampled permitted orders. A submitted order may pass even when another permitted interleaving fails.

Before execution, the program must be well formed. The parser requires JSON in the last `<answer>` segment, all seven fields, non-empty unique step triples, valid references and correctly shaped objects, bindings and states. Trajectories have 2–64 six-field rows (expanded to eight before parsing), with non-decreasing progress from 0 to 1. It applies these checks before scoring and returns -1 on parse failure.

K0 rejects self-loops, duplicate edges, cycles over all steps, repeated/unknown IDs and listed-order edge violations; controllers may be absent from the order. Its verdict applies after scoring and replay. Algorithm 1 specifies exception handling.

The symbolic engine tests action legality from reference initial facts under fixed operator rules, rebuilding arguments from roles, bindings, and moving entities. Legal execution requires every precondition and hand rule to pass and the final hands to be empty. Goal achievement separately checks the reference outcome facts (Table 10).

Table 10: **Execution state sources and denominators.** EXEC-1 and EXEC-F average sampled symbolic legal-order shares; repeated orders count.

Readout	State source / orders	Quantity / denominator	Population / use
Declared-state replay			
Witness W	Declared states and `init` facts; submitted $\hat{L}$	Boolean of Eq. (15)	Witness pass rate; training hard gate
EXEC-1, $1 - \widehat{\ell}$	Same declared states; 20 seeded ready-node orders of $E\|_{\hat{L}}$	Legal share over sampled orders; the separate witness is excluded from the denominator	All items; parse failure 0. Excess risk enters reward
Independent symbolic engine			
EXEC-F	Reference initial state and fixed operator rules; 50 random orders over non-controller steps	Legal share: all preconditions and one-hand rules pass; nothing remains grasped	All items; parse failure 0. Evaluation only (Figure 8c)
Goal achievement	Same 50 engine runs; goals = reference outcome facts minus initial facts	Share ending with every goal fact; legality is not required	Parse-valid items; evaluation only

In Figure 10, grasping establishes the hand fact used by the tool actions, and release removes it. Moving an independent step changes the sequence without breaking this support chain. The reordered sequence remains legal because each tool action retains the state support established by grasping.

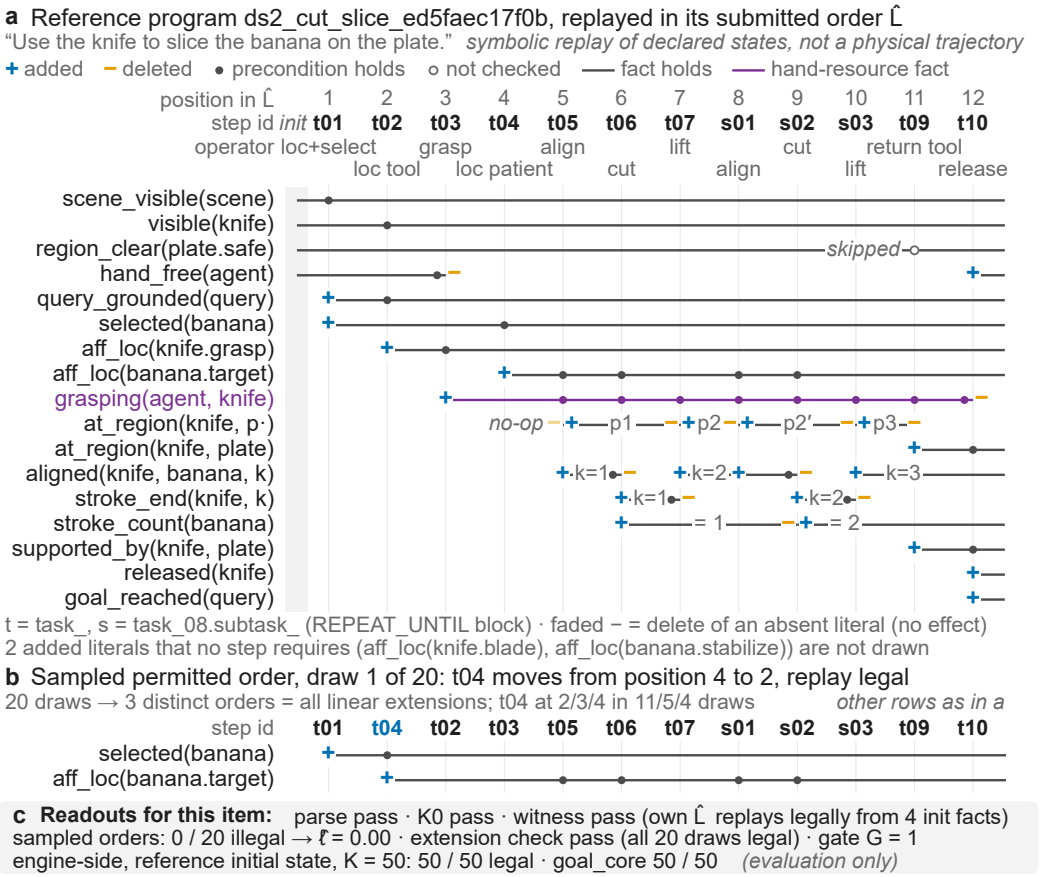


Figure 10: **One real replay under declared states.** The reference program of the Figure 3 task, replayed symbolically in its submitted order (a). Columns are steps, rows declared literals; lines mark facts holding, +/− additions/deletions, filled dots checked preconditions, the hollow dot the exempt `region_clear` test. The hand fact (violet) is added at t03, required by every tool step, deleted at t10. (b) The first sampled order moves only t04 and stays legal; (c) this item's readouts (Table 10).

C.2 How do the execution checks enter the reward?

The training reward combines the complete relation score with these execution checks. Complete heads are clipped to $[0, 1]$ ($q_h^{\mathrm{cc}} = q_h^{\mathrm{r}}$ in the main text), and the final run enables the structural and witness gates:

$$M_{\mathrm{cc}}(\hat{P}, P^\star) = \frac{1}{|\mathcal{H}(P^\star)|} \sum_{h \in \mathcal{H}(P^\star)} q_h^{\mathrm{cc}}(\hat{P}, P^\star), \qquad S = 2M_{\mathrm{cc}} - 1,$$
$$G(\hat{P}) = \mathrm{parse}(\hat{P}) \wedge \mathrm{K0}(\hat{P}) \wedge W(\hat{P}). \tag{16}$$

A permitted reference order can itself interleave grasp blocks and fail the hand rule. We therefore compare the prediction with the reference under identical sampling and discount only the prediction's excess failure share:

$$e(\hat{P}, P^\star) = \big[\widehat{\ell}(\hat{P}) - \widehat{\ell}(P^\star)\big]_+, \qquad \beta \geq 0,$$
$$R_{\mathrm{cc}}(\hat{P}, P^\star) = \begin{cases} -1, & G(\hat{P}) = 0, \\ \mathrm{clip}\big(2M_{\mathrm{cc}}(1 - \beta e) - 1,\ -1,\ 1\big) = \mathrm{clip}\big(S - \beta e\,(S+1),\ -1,\ 1\big), & G(\hat{P}) = 1, \end{cases} \tag{17}$$

Each draw chooses uniformly among the ready steps of $\hat{L}$ under $E|_{\hat{L}}$, and the same sampling rule is used for the prediction and reference. We use $K = 20$ draws; $\widehat{\ell}$ is the fraction of completed

draws with a failing step, with undefined cycles assigned failure. Repeated orders count, and the submitted-order witness is checked separately.

Property (excess risk never raises the reward). Fix $M_{\mathrm{cc}} \in [0,1]$, $G(\hat{P}) = 1$, $\widehat{\ell}(P^\star)$, and $\beta \geq 0$. Then $e \mapsto R_{\mathrm{cc}}$ is non-increasing on $[0,1]$, equals S at $e = 0$, and equals 1 when every active head is 1 and $e = 0$.

Proof. The unclipped slope is $-\beta(S+1) = -2\beta M_{\mathrm{cc}} \leq 0$; monotone clipping can add flat segments (inactive for $0 \leq \beta \leq 1$). Identical reference draws give zero excess risk; with all heads 1, the reward is 1. The final run uses $\beta = 1$. □

Algorithm 1 Complete training reward ($\beta = 1$, $K = 20$).

1: **Input:** response x, reference program; structural and witness gates on, $\beta = 1$, $K = 20$; initialize `overall`$=-1$.
2: Load the reference, masks, active heads, comparabilities, and its sampled failure share (undefined $\to 1$) under the same sampling protocol used for the prediction.
3: Parse the last `<answer>` segment; **if** rejected, **return** -1.
4: Set $k_0 = 1$ and run K0 if enabled. A checker exception leaves $k_0 = 1$; a negative verdict sets $k_0 = 0$.
5: Compute the shared matching, heads, witness and sampled failure share as specified in Appendix C.1 and below Eq. (17); undefined prediction share $\to 1$.
6: If an enabled K0 or witness gate fails, return -1; otherwise return Eq. (17).
7: Any other exception in lines 2–6 returns the default -1.

C.3 What reaches the optimizer?

GRPO receives only the scalar `overall`; the head values, gates, and risk terms are diagnostics rather than additional training inputs. CBA is evaluation-only, and the same reward calculation is used for every item (Appendix D).

D Experimental and Reproduction Protocols

D.1 What is held fixed across comparisons?

All comparisons use the same train–calibration–test split. GRPO draws prompts from training, while development and calibration select checkpoints. Offline design analyses reuse the same tasks and saved A0 candidates (Table 11).

The tables distinguish the supervised starting point (A0), original-reward training (P6), complete-reward training (R_{cc}), and a flat action-sequence target (N-1). Dependency reconstruction (X1) edits a fixed output; structural selection (SEL8) chooses among samples without training.

Supervised starting point. A0 starts from Embodied-R1.5-8B-SFT (Yuan et al., 2026). LoRA uses rank 64, $\alpha = 128$, dropout 0.05 on language-model linear layers, with the vision tower and projector frozen. SFT-1 trains on record fragments for three epochs (lr 10^{-4}, cosine schedule, 16 sequences/update); SFT-2 continues the adapter on whole programs for two epochs (lr 5×10^{-5}, 8 programs/update). Both use cutoff 16,384, bf16, and DeepSpeed ZeRO-2; the final adapter is merged for decoding.

Common decoding and reward-training recipe. The reward-training and checkpoint-selection settings below describe the seed-42 comparison. A0-lineage arms use one greedy decode with vLLM 0.11.0, temperature 0, top-p 1, tensor parallelism 1, batch 96, and output/context limits 12,288/20,480 tokens, except E′ (Table 11). Both GRPO runs use EasyR1, seed 42, 32 prompts × 8 samples/update, temperature and top-p 1, prompt/response limits 4,096/12,288, actor lr 3×10^{-6} (AdamW, constant), clip 0.2/0.28, one PPO epoch, zero KL coefficient, and a frozen vision tower. Checkpoints and validation occur every eight updates.

Table 11: **Training and processing configurations.** All test decodes are frozen; "same data" denotes the 1,254/84/360 split above. The table gives the settings needed to reproduce each comparison.

Arm	Start / intervention	Data / budget / selection
Supervised controls (Table 13)		
A0	Embodied-R1.5-8B-SFT; seven-key program target, compact v3 prompt	Same data; 3+2 epochs, 36.1 GPU·h on four GPUs. Final adapter; seeds 7/123 repeat SFT-2 only.
N-1	A0 SFT-1 checkpoint; flat action-sequence target in SFT-2. A training-derived lifter supplies states and chain dependencies.	Same 1,254 records; two epochs, 314 updates; batch 8, lr 5×10^{-5}, seed 42. Matches A0 SFT-2 budget.
B-0	Qwen3-VL-8B-Instruct; otherwise A0 recipe	Same data, 3+2 epochs; seeds 42/7/123.
Other backbones	Qwen3-VL-2B/4B, InternVL3-8B, GLM-4.1V-9B-Thinking; A0 recipe	Same data, 3+2 epochs. 2B/4B: two GPUs, accumulation 8.
A0 re-decode	Same A0 checkpoint, repeated standard decode	Test 360; no training; decode-noise control.
Reward intervention (Figure 8; Table 17)		
P6	A0 merged weights; original sequence reward R_{p1} behind K0	1,000 training prompts; dev/calib selection. *Selected:* update 32, 8,192 update rollouts. *Configured:* 40 updates; length guard stopped the run. 26.2 GPU·h.
$\boldsymbol{R_{\mathrm{cc}}}$	A0 merged weights; Eq. (4), v2.2, $\beta = 1$, $K = 20$, structural and witness gates	Same data. *Selected:* update 32, 8,192 update rollouts. *Configured and completed:* 32 updates. ≈20.9 GPU·h.
Other post-training arms (Figure 7)		
L0	P6 update 32; hard replay gate (any of 20 sampled orders illegal ⇒ -1)	Same GRPO split; five further updates. Selection gate failed; forced test decode.
RFT-1	A0 SFT-2 adapter; eight samples ($T = 0.7$, top-$p = 0.95$) plus greedy, K0 and v1 selection, X1-reencoded targets	Train 1,254, calib 84 gate; 10,032 samples; one epoch at lr 5×10^{-5}, 157 updates, 10.4 GPU·h.
RFT-2	RFT-1; second round, cumulative targets	Same data and recorded recipe.
GLM-RFT	GLM model trained with the A0 recipe; RFT-1 procedure	Same RFT-1 data; one round.

Frozen program processing and supervised continuations.

Arm	Start / intervention	Data / completed budget
Frozen-output processing		
X1	A0 frozen output; rebuild dependencies from declared providers, controllers, and threat protection. Steps, order, states, and motion unchanged.	Test 360; no training; applied to seeds 42/7/123.
SEL8	A0; reference-free structural selection among eight sampled programs	Test 360; eight samples/item, no training.
full	SEL8 followed by X1	Same test candidates, no training.
Supervised continuations		
G0	RFT-2 adapter; reference-target SFT, uniform token weights	Train 1,254; one epoch, lr 5×10^{-5}, batch 8; 157 updates, 1,717 s on four GPUs.
G1	RFT-2 adapter; G0 with 4:1 weights on elements marked wrong or missing by a frozen matcher	Same data and budget as G0.
E′	RFT-2 adapter; draft/fix/answer repair targets	2,174 repair rows from training; one S3″ SFT run. Test output limit 20,480 (extended).

Reward selection and stopping. Both reward runs select among updates 8/16/24/32 using the original accounting on calib-84 (ties use dev-254 and then original order score), while requiring calibration and development scores at least at initialization. Both select update 32. P6 has a 40-update budget and stops after mean rollout length exceeds 1.3 times the A0 mean in two consecutive batches; the complete-reward run completes its fixed 32-update budget with the structural and witness gates active. The two selected checkpoints therefore use the same 8,192 update-participating rollouts.

Reference-free candidate selection. SEL8 selects programs using reference-free criteria for readability and dependency structure. It chooses from the first eight of 16 samples by the lexicographic tuple $(p, d, c, t, -r, -i)$: readability, acyclic/order-consistent graph, an incoming edge after the first step, non-total order, fewer redundant edges, and then sample index. The same rule is applied to every candidate set.

Model sources. The backbone and external comparisons use Qwen3-VL (Bai et al., 2025a), Qwen2.5-VL (Bai et al., 2025b), InternVL3 (Zhu et al., 2025), GLM-4.1V-Thinking (V Team et al., 2025), Gemma 3 (Gemma Team et al., 2025), Llama 3.2 Vision (Meta, 2024), and MiniCPM-V 4.5 (Yu et al., 2025b). Sizes and prompts appear in Table 14.

External prompts. The external comparison varies what the prompt asks the model to express. A0 uses the training prompt; E-1b omits states; E-1d supplies the seven-key program with worked examples; E-1c/G and E-1dG add the corresponding schema or grammar constraint. The protocol table specifies the requested fields, worked examples, token budgets, and grammar constraints for each comparison. Token budgets are output/context limits.

External-model prompt protocols

Protocol	Requested content / examples	Token budget / grammar
E-1b	Objects, steps, deps, order, regions, trajectories; bounding boxes, **no states**; zero-shot	4,096/12,288; none
E-1d	Seven-key program, **required states**; three fixed non-test worked examples, split-checked	16,384/32,768; none. Llama output: 4,096
E-1c / G	Seven-key schema without examples; E-1c-G adds xgrammar	Original-scorer results only
E-1dG	E-1d prompt and examples, xgrammar JSON schema	E-1d budget; original-scorer results only

All external arms decode once greedily on test 360. Llama-3.2-11B uses a 4,096-token output limit, and GLM reasoning text is stripped before parsing; grammar constraints are applied after reasoning. The protocol table and Appendix A fix the requested fields and parsing rules.

D.2 How are the reported differences computed?

Unless a row says original, all relation scores use commitment-complete accounting; CBA follows Appendix B.3. All-item results cover 360 tasks: parse failures contribute 0 to unsigned heads and legality, -1 to signed scores, and 0.5 to contrastive pairs. Parse-valid levels use each arm's printed n; paired heads use jointly parsed tasks. Coverage and conditional correctness use the stated pooled or task-mean aggregation; the shared-pair analysis reports both (Appendix G.2).

Unless noted otherwise, intervals are nominal 95% item-level percentile bootstraps with $B = 5{,}000$; resampling preserves item pairing. A star marks an interval excluding zero. The statistical units and populations for each comparison are listed in Table 12.

Table 12: **Populations and statistical units.** The text specifies shared failure and bootstrap rules; rows list exceptions and the exact comparison. Construction and endpoint levels without intervals are descriptive.

Result group	Population / comparison	Intervals / limits
Constructions (Table 9)	Nine deterministic rewrites versus reference; all 360 tasks. Direct score $2M_{\text{cc}} - 1$ without gate/risk; CBA balanced by family.	$B = 5{,}000$ for paired four+deletion CBA differences; no level intervals. Italics include zero. †: within 11 A0 re-decodes' band (max $\lvert\Delta\rvert = 0.004$, half-width ≤ 0.012).
Controls (Table 4)	N-1/B-0 versus A0: CBA on 360; paired ORD3 on 327/315 jointly parsed tasks. Levels and coverage: 333/335/328 parsed tasks for A0/N-1/B-0.	Paired $B = 5{,}000$; pooled coverage has no interval here. Table 13 gives conditional differences ($B = 10{,}000$) and their separate n.
Rescoring (Fig. 7a)	Twelve arms versus A0; all 360 tasks. Each frozen output is scored separately under both versions.	$B = 5{,}000$; marks compare significance status. G0/G1/E$'$ start from RFT-2; E$'$ uses 20,480 output tokens.
Conditional order (Fig. 7b)	Five v1-trained arms versus A0 on 324–329 jointly parsed tasks expressing a pair; task-equal mean of within-item conditional-ORD3 differences.	Conditional-correctness intervals use 10,000 paired task resamples; contrast-accuracy intervals use 5,000, as in panel (a).
Program contents (Fig. 7c)	Padding/dependencies: own parsed tasks; self-replay and engine legality: all 360, parse failure 0.	Descriptive item means; — means not recorded. Replay sample sizes below.
Reward (Fig. 8, Table 17)	*Curves:* dev-254, updates 0/8/16/24/32; padding on parsed outputs, replay on all items. *Test:* 360; parsed A0/P6/new 333/349/356; jointly parsed 333 vs A0, 348 vs P6.	Curves: no level intervals or intermediate test readings. Endpoints: $B = 5{,}000$; goal uses parsed tasks, deletion CBA its own family. This row describes seed 42; Table 18 reports seeds 7 and 123.

EXEC-1 uses 20 orders and EXEC-F 50, as specified in Appendix C.1. Test outputs are frozen before scoring.

E Extended Results on Task Commitments

Accurate expressed relations can coexist with an incomplete task. The supervision controls and external-model comparisons separate these two effects by reading program-level agreement alongside coverage and conditional correctness.

E.1 Does joint supervision change what is expressed?

At comparable step and pair coverage, joint program supervision improves contrast accuracy and three-way order agreement over the flat action-sequence target (Table 13). The confidence interval for the paired difference in correctness on expressed relations spans zero.

The backbone controls also expose the role of parsing. Qwen3-VL and GLM obtain similar results with the same supervision. InternVL3's scores on its parsed subset look comparable, but fewer outputs parse and its paired relation agreement is lower. Comparing only each model's successful outputs would hide this difference in which tasks reach evaluation.

Table 13: **Joint supervision improves relation agreement at similar coverage.** Blocks (a)–(c) report levels, paired differences from the program-supervised baseline, and individual heads on the 360 test tasks. Panel A changes the supervision target or seed; panel B changes the backbone. Bold links the program–sequence comparison across coverage and relation scores. Shared evaluation conventions are in Appendix D.2.

(a) Levels

ID	Target / backbone	n_{pv}	CBA	ORD3	Step cov.	Pair cov.	Cond. corr. [95% CI]	EXEC-1
Panel A: supervision target; backbone Embodied-R1.5-8B-SFT								
A0	**Joint program**, seed 42	333	**0.828**	**0.465**	0.731	**0.362**	0.898 [0.879, 0.915]	0.890
A0-s7	seed 7	330	0.816	0.472	0.734	0.373	0.903 [0.887, 0.919]	0.902
A0-s123	seed 123	328	0.826	0.468	0.730	0.358	0.902 [0.884, 0.919]	0.895
N-1	**Flat sequence**, lifted	335	**0.792**	**0.439**	0.729	**0.357**	0.869 [0.848, 0.890]	0.186‡
Panel B: backbone; same joint program target and recipe								
B-0	Qwen3-VL-8B-Instruct	328	0.826	0.465	0.737	0.352	0.893 [0.868, 0.914]	0.874
B-G	GLM-4.1V-9B-Thinking	328	0.823	0.466	0.735	0.370	0.905 [0.888, 0.920]	0.883
B-I	InternVL3-8B	288	0.794	0.463	0.751	0.368	0.908 [0.891, 0.924]	0.759

(b) Paired differences from joint program supervision (row − A0)

ID	ΔCBA [95% CI]	ΔORD3 [95% CI]	n	Δq_E [95% CI]	Δ cond. corr. [95% CI]	n
A0-s7	-0.011 $[-0.025, +0.002]$	$+0.006$ $[-0.009, +0.022]$	323	$+0.003$ $[-0.016, +0.021]$	—	—
A0-s123	-0.002 $[-0.015, +0.012]$	-0.004 $[-0.017, +0.009]$	321	-0.005 $[-0.022, +0.012]$	—	—
N-1	$\mathbf{-0.036}$ $[-0.055, -0.018]^*$	$\mathbf{-0.025}$ $[-0.043, -0.008]^*$	327	-0.015 $[-0.034, +0.003]$	$\mathbf{-0.001}$ $[-0.024, +0.023]$	322
B-0	-0.001 $[-0.019, +0.016]$	-0.002 $[-0.017, +0.015]$	315	-0.006 $[-0.025, +0.013]$	$+0.000$ $[-0.010, +0.010]$	313
B-G	-0.005 $[-0.025, +0.015]$	-0.008 $[-0.027, +0.011]$	312	-0.004 $[-0.025, +0.019]$	—	—
B-I	-0.034 $[-0.056, -0.012]^*$	-0.044 $[-0.065, -0.024]^*$	280	-0.062 $[-0.085, -0.039]^*$	—	—

(c) Commitment-complete heads, means over parse-valid items

ID	q_O	q_S	q_B	q_E	q_C	q_T	q_F
A0	0.586	0.701	0.436	0.508	0.516	0.318	0.015
A0-s7	0.587	0.704	0.450	0.516	0.519	0.309	0.016
A0-s123	0.591	0.701	0.443	0.510	0.519	0.316	0.020
N-1	0.593	0.681	0.434	0.488	0.467‡	0.314	0.018
B-0	0.603	0.703	0.445	0.509	0.519	0.316	0.018
B-G	0.599	0.698	0.451	0.512	0.510	0.289	0.012
B-I	0.615	0.712	0.470	0.510	0.529	0.301	0.013

CBA and EXEC-1 use all 360 tasks; ORD3 and individual heads use each model's n_{pv} parsed tasks.Paired ORD3 differences use the first n, whereas paired q_E differences use all 360 tasks with parse failures scored 0; conditional-correctness differences use the last n, where both models express a pair. These paired differences need not equal differences between pooled levels. Intervals use 5,000 task resamples, or 10,000 for conditional correctness. EXEC-1 averages 20 replays without an interval. The flat sequence is lifted with step IDs, chain dependencies, and template states; ‡ marks these supplied states rather than learned content. Metric definitions: Appendices B.1 and C.1.

E.2 What do external models leave unexpressed?

Among three-shot models with broad parsing success, Qwen2.5-VL-32B and InternVL3-38B describe expressed relations accurately but cover fewer required steps and pairs than the program-supervised model (Table 14). Qwen's higher conditional correctness on shared items coexists with substantially lower pair coverage. Correctness on the relations a model chooses to express therefore does not establish that it has described the whole task.

Table 14: **Accurate expressed relations can leave much of the task uncovered.** Panel (a) reports complete relation scores; panel (b) separates coverage from conditional correctness. Z rows use zero-shot prompts, S rows three examples, and R the supervised model's training prompt. Bold connects Qwen2.5-VL-32B's lower coverage with its higher paired conditional correctness. Prompt and evaluation conventions: Appendix D.2.

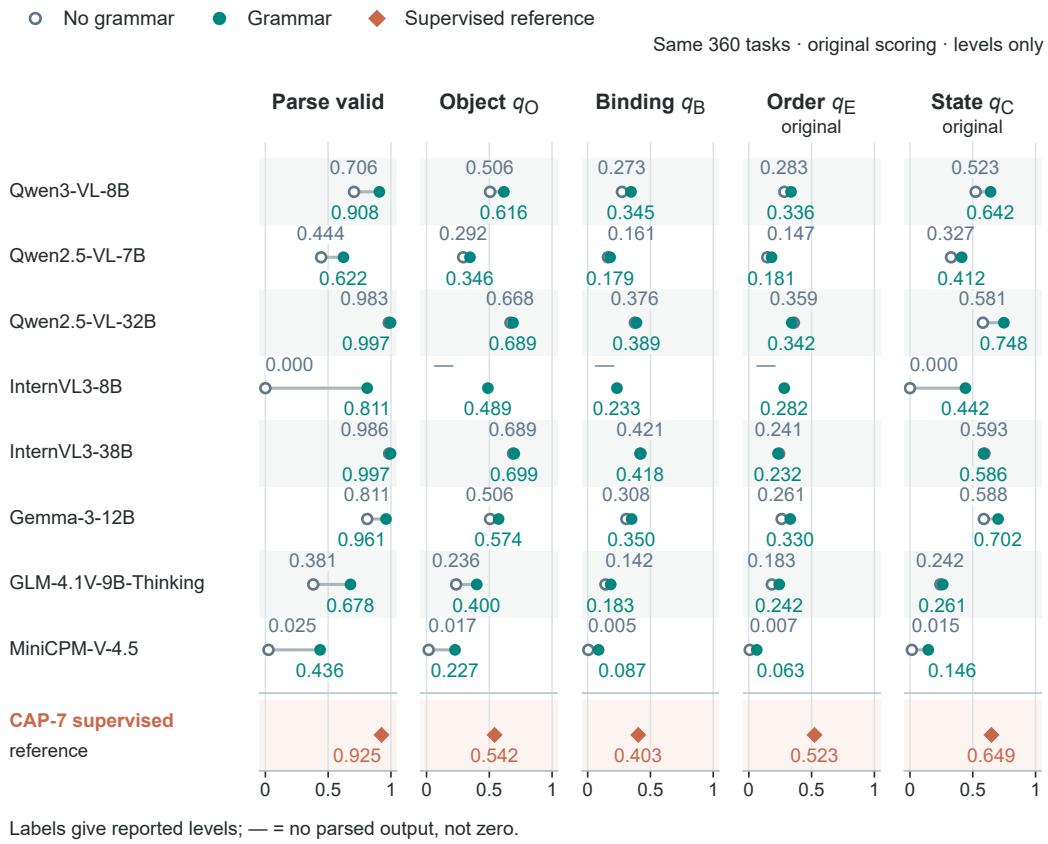
(a) Commitment-complete heads and CBA, hard gate, 360 items

ID	Model	q_O	q_S	q_B	q_E	q_C	q_T	q_F	CBA	ΔCBA vs. R [95% CI]
Zero-shot, E-1b (compact schema, no state field)[†]										
Z1	Qwen3-VL-8B	0.469	0.474	0.000	0.139	0.000	0.063	0.007	0.711	-0.117 $[-0.144, -0.088]^*$
Z2	Qwen2.5-VL-7B	0.354	0.358	0.000	0.065	0.000	0.076	0.006	0.666	-0.162 $[-0.191, -0.132]^*$
Z3	Qwen2.5-VL-32B	0.458	0.406	0.000	0.085	0.000	0.094	0.012	0.736	-0.092 $[-0.123, -0.061]^*$
Z4	InternVL3-8B	0.483	0.395	0.000	0.085	0.000	0.077	0.012	0.699	-0.129 $[-0.159, -0.099]^*$
Z5	InternVL3-38B	0.459	0.434	0.000	0.107	0.000	0.082	0.013	0.710	-0.118 $[-0.148, -0.086]^*$
Z6	Gemma-3-12B	0.406	0.422	0.000	0.100	0.000	0.083	0.009	0.704	-0.124 $[-0.156, -0.093]^*$
Z7	GLM-4.1V-9B-Thinking	0.307	0.332	0.000	0.062	0.000	0.035	0.023	0.610	-0.218 $[-0.245, -0.190]^*$
Z8	Llama-3.2-11B-Vision	0.103	0.190	0.000	0.009	0.000	0.019	0.000	0.542	-0.286 $[-0.314, -0.258]^*$
Z9	MiniCPM-V-4.5	0.342	0.377	0.000	0.082	0.000	0.041	0.007	0.631	-0.197 $[-0.225, -0.169]^*$
Z10	RoboBrain2.0-7B	0.407	0.400	0.000	0.103	0.000	0.071	0.006	0.659	-0.169 $[-0.198, -0.140]^*$
Z11	Embodied-R1.5-8B-SFT[♭]	0.494	0.203	0.000	0.021	0.000	0.003	0.000	0.522	-0.306 $[-0.330, -0.282]^*$
3-shot, E-1d16k (full schema, three worked examples)										
S1	Qwen3-VL-8B	0.506	0.415	0.273	0.221	0.319	0.148	0.004	0.738	-0.090 $[-0.117, -0.062]^*$
S2	Qwen2.5-VL-7B	0.292	0.280	0.161	0.161	0.214	0.099	0.001	0.660	-0.168 $[-0.196, -0.140]^*$
S3	Qwen2.5-VL-32B	0.668	0.592	0.376	0.307	0.358	0.179	0.004	0.796	-0.031 $[-0.056, -0.007]^*$
S4	InternVL3-8B[¶]	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.500	-0.328 $[-0.351, -0.304]^*$
S5	InternVL3-38B	0.689	0.582	0.421	0.300	0.357	0.168	0.003	0.808	-0.019 $[-0.045, +0.006]$
S6	Gemma-3-12B	0.506	0.479	0.308	0.248	0.365	0.139	0.005	0.760	-0.068 $[-0.096, -0.039]^*$
S7	GLM-4.1V-9B-Thinking	0.236	0.251	0.142	0.151	0.165	0.035	0.000	0.631	-0.196 $[-0.223, -0.169]^*$
S8	Llama-3.2-11B-Vision[§]	0.062	0.077	0.041	0.046	0.048	0.018	0.000	0.535	-0.292 $[-0.318, -0.267]^*$
S9	MiniCPM-V-4.5	0.017	0.017	0.005	0.009	0.010	0.000	0.000	0.510	-0.318 $[-0.341, -0.294]^*$
Jointly supervised CAP-7 reference, training prompt										
R	CAP-7 supervised	0.542	0.648	0.403	0.470	0.477	0.294	0.014	0.828	—

(b) Parse-valid items: order, coverage, and conditional correctness

ID	n_{pv}	Parse-valid	ORD3	Step cov.	Pair cov.	Cond. corr. [95% CI]	Δ cond. corr. vs. R [95% CI] (n)
Zero-shot; model IDs as in panel (a)							
Z1	360	1.000	0.119	0.400	0.141	0.723 [0.670, 0.773]	—
Z2	360	1.000	0.064	0.275	0.068	0.788 [0.747, 0.826]	—
Z3	360	1.000	0.083	0.321	0.085	0.800 [0.760, 0.838]	—
Z4	360	1.000	0.070	0.340	0.088	0.622 [0.580, 0.664]	—
Z5	360	1.000	0.104	0.328	0.094	0.873 [0.847, 0.897]	—
Z6	360	1.000	0.102	0.356	0.092	0.879 [0.851, 0.904]	—
Z7	359	0.997	0.054	0.256	0.066	0.724 [0.664, 0.782]	—
Z8	336	0.933	0.010	0.127	0.013	0.641 [0.541, 0.751]	—
Z9	360	1.000	0.068	0.319	0.099	0.637 [0.587, 0.686]	—
Z10	360	1.000	0.087	0.359	0.100	0.776 [0.751, 0.799]	—
Z11	360	1.000	0.018	0.130	0.020	0.851 [0.793, 0.897]	—
Three-shot; model IDs as in panel (a)							
S1	254	0.706	0.284	0.573	0.141	0.896 [0.869, 0.921]	$+0.024$ $[-0.004, +0.054]$ (231)
S2	160	0.444	0.337	0.570	0.269	0.926 [0.886, 0.956]	—
S3	354	0.983	0.281	0.564	**0.196**	0.888 [0.864, 0.911]	$\mathbf{+0.027}$ $[+0.004, +0.050]^*$ (325)
S4	0	0.000	—	—	—	—	—
S5	355	0.986	0.271	0.559	0.167	0.870 [0.848, 0.892]	$+0.022$ $[-0.003, +0.046]$ (322)
S6	292	0.811	0.269	0.557	0.200	0.879 [0.853, 0.903]	$+0.021$ $[-0.004, +0.047]$ (273)
S7	137	0.381	0.373	0.623	0.318	0.946 [0.920, 0.967]	—
S8	44	0.122	0.354	0.593	0.282	0.932 [0.882, 0.969]	—
S9	9	0.025	0.315	0.642	0.356	0.836 [0.711, 0.916]	—
Jointly supervised CAP-7 reference							
R	333	0.925	0.465	0.731	**0.362**	0.898 [0.879, 0.915]	—

Panel (a) uses all 360 tasks; parse failures score 0 for heads and 0.5 for CBA. Panel (b) uses each model's parsed tasks; paired conditional differences use the printed shared n, not differences of pooled levels. CBA/conditional intervals use 5,000/10,000 resamples. [†] Zero-shot prompts request neither states nor matching region IDs, explaining zero q_C/q_B. [§] Llama uses 4,096 output tokens; other three-shot runs use 16,384/32,768 output/context tokens. [¶] No three-shot output parses, leaving conditional scores undefined. [♭] Untuned backbone of the supervised model. Full settings: Appendix D.1.



Labels give reported levels; — = no parsed output, not zero.

Figure 11: **Grammar constraints improve parsing without uniformly improving relations.** Paired points show three-shot decoding without and with xgrammar on the same 360 test tasks; the separate final row is the supervised reference. Both conditions use the original scorer: q_O and q_B are unchanged across scorers, whereas q_E and q_C retain their original definitions. Endpoint labels include reported zeros; dashes denote undefined readouts from the unparsed run.

xgrammar uses the same prompt; GLM applies it after reasoning. Qwen3-VL-8B, Qwen2.5-VL-7B/32B, and InternVL3-8B constrained runs use different machines from their unconstrained runs, so only levels are reported, without paired intervals. Llama has no constrained run. The original protocol uses a hard gate, scoring parse failures 0; the unparsed InternVL3-8B run nevertheless reports $q_O/q_B/q_E$ as undefined, retained here as dashes.

Grammar constraints raise parsing success in every tested model, while order and binding can remain flat or decline (Figure 11). The separation is clearest when parsing is already near its ceiling: constraining the serialization does not resolve which task relations the model chooses to express.

The examples in Figure 12 locate the coverage gap within otherwise plausible plans. Missing intermediate actions break a sequence even when the named tool and target are appropriate; a wrong binding instead attaches an available action to the wrong region. Joint supervision connects more of these steps, making the coverage difference visible as program structure.



Figure 12: **Joint supervision connects more of the shared task.** Six tasks compare supervised and external-model programs with their references. Colours show predicted roles on reference regions; markers locate missing steps, wrong bindings, and differing prerequisite sources. External models use the zero-shot schema, which requests no states; missing-state labels therefore reflect that protocol.

F Rescoring the Same Programs and Explaining Their Changes

F.1 Which gains survive complete rescoring?

Rescoring holds the generated plans fixed, so changes in the measured gains arise from the evaluator's scoring and comparison rules. This isolates whether improvements under the original verifier survive checks for omitted obligations and unsupported additions (Table 15).

The original-reward run, replay-gated continuation, and selected-sample fine-tuning all optimize against the earlier verifier. Dependency reconstruction changes only the written edges; structural selection chooses an existing candidate. These comparisons trace dependence on the scoring rule across training, sample selection, and dependency reconstruction.

The complete evaluation protocol reduces the apparent gains of original-score training without changing a single prediction. The remaining contrast gains do not consistently imply better order agreement. Adding deletion contrasts further changes the interpretation of selected-sample fine-tuning (Figure 13): improvements against the four modification families largely disappear once omitted requirements are tested. Dependency reconstruction likewise changes from a small positive difference to a negative one under the expanded comparison. The choice of alternatives therefore matters even after the heads have been corrected.

Table 15: **The scorer changes which gains remain.** The same programs are read with the original and complete scorers. Panel (a) gives levels; panel (b) gives paired gains from A0. Bold entries mark gains whose interpretation changes after required relations are retained.

(a) Objects and absolute readouts.

Arm	Start / operation	Base	CBA Orig.	CBA Comp.	ORD3	parse	EXEC-1	n_{pv}
Baseline								
A0	two-stage SFT, seed 42	—	0.707	0.828	0.465	0.925	0.890	333
RL against the original verifier								
P6	GRPO, original reward, 32 updates	A0	0.772	0.846	0.411	0.969	0.553	349
L0	P6 + hard replay gate, 5 more steps	A0	0.765	0.848	0.434	0.978	0.666	352
Selection-then-SFT (RFT) against the original verifier								
RFT-1	SFT on original-score selections	A0	0.737	0.835	0.449	0.933	0.897	336
RFT-2	second RFT round, from RFT-1	A0	0.741	0.842	0.437	0.953	0.916	343
RFT-new	third round, new-scene items added	A0	0.749	0.840	0.431	0.981	0.955	353
GLM-RFT	RFT-1 recipe on GLM-4.1V-9B	A0	0.731	0.835	0.433	0.928	0.880	334
GLM-FT	A0 recipe on GLM-4.1V-9B (level row)	A0	0.681	0.818	0.464	0.908	0.880	327
Inference-time operations on A0 outputs								
X1	A0 outputs, dependencies re-encoded	A0	0.735	0.834	0.443	0.925	0.894	333
SEL8	eight A0 samples, structural selector	A0	0.723	0.829	0.414	0.997	0.946	359
full	SEL8 outputs re-encoded as X1	A0	0.733	0.834	0.386	0.997	0.967	359
Supervised continuations from RFT-2								
G0	reference-target SFT from RFT-2, uniform	A0	0.692	0.817	0.483	0.917	0.905	330
G1	as G0, span-weighted	A0	0.607	0.801	0.496	0.900	0.875	324
E′	self-repair SFT from RFT-2	A0	0.720	0.809	0.343	0.978	0.932	352

(b) Paired gains against the baseline.

Arm	ΔCBA original [95% CI]	ΔCBA complete [95% CI]	Original → complete	ΔORD3 [95% CI]	$n_{\Delta\mathrm{ORD3}}$
RL against the original verifier					
P6	**+0.065** [+0.037, +0.093]*	**+0.018** [−0.001, +0.037]	* → n.s.	−0.048 [−0.068, −0.030]*	331
L0	**+0.059** [+0.031, +0.087]*	**+0.020** [+0.002, +0.038]*	* → *	−0.022 [−0.040, −0.004]*	331
Selection-then-SFT (RFT) against the original verifier					
RFT-1	**+0.030** [+0.010, +0.051]*	**+0.007** [−0.007, +0.022]	* → n.s.	−0.022 [−0.039, −0.005]*	328
RFT-2	**+0.034** [+0.012, +0.056]*	**+0.014** [−0.002, +0.031]	* → n.s.	−0.023 [−0.040, −0.006]*	327
RFT-new	**+0.042** [+0.019, +0.066]*	**+0.013** [−0.004, +0.029]	* → n.s.	−0.022 [−0.038, −0.007]*	332
GLM-RFT	+0.024 [−0.005, +0.055]	+0.007 [−0.015, +0.029]	n.s. → n.s.	−0.036 [−0.054, −0.018]*	317
GLM-FT	−0.025 [−0.054, +0.004]	−0.010 [−0.032, +0.011]	n.s. → n.s.	−0.007 [−0.029, +0.013]	312
Inference-time operations on A0 outputs					
X1	+0.028 [+0.016, +0.042]*	+0.006 [+0.001, +0.012]*	* → *	−0.022 [−0.028, −0.017]*	333
SEL8	+0.016 [−0.011, +0.044]	+0.002 [−0.020, +0.022]	n.s. → n.s.	−0.042 [−0.061, −0.022]*	332
full	+0.026 [+0.001, +0.053]*	+0.006 [−0.014, +0.027]	* → n.s.	−0.069 [−0.088, −0.050]*	332
Supervised continuations from RFT-2					
G0	−0.015 [−0.038, +0.009]	−0.011 [−0.029, +0.007]	n.s. → n.s.	+0.011 [−0.007, +0.029]	317
G1	−0.100 [−0.128, −0.070]*	−0.027 [−0.047, −0.007]*	* → *	+0.020 [−0.005, +0.044]	319
E′	+0.013 [−0.016, +0.042]	−0.019 [−0.040, +0.002]	n.s. → n.s.	−0.114 [−0.134, −0.092]*	332

Original/complete denote the original and commitment-complete scorers. CBA is averaged over the applicable corruption families; parse failures score 0.5. ORD3 counts agreement on reference and unmatched pairs for parse-valid programs. EXEC-1 is legality over 20 declared-state replays. Stars mark intervals excluding zero; n.s. includes zero; — is unpaired. GLM-FT is the same-backbone start of GLM-RFT, and E′ uses a larger decoding budget, so both are level rows. Full settings: Appendix D.1.

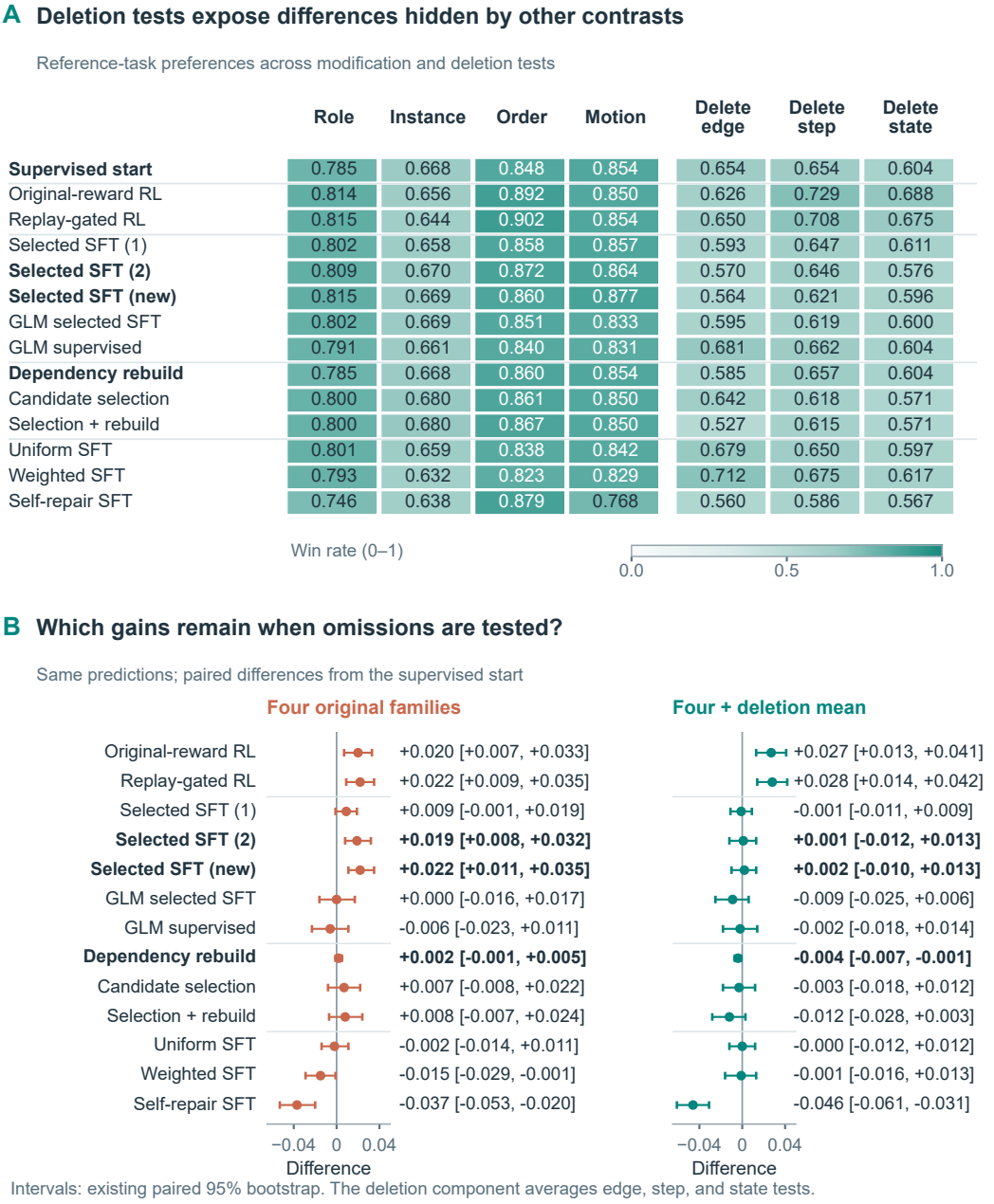


Figure 13: **Testing omitted requirements changes which gains remain.** (A) Complete-scoring win rates for four modification and three deletion families. (B) Paired gains over the supervised start across 360 tasks. Selected-sample fine-tuning can improve modification contrasts without improving the expanded comparison. Bold marks the two later fine-tuning rounds and dependency reconstruction, whose interpretation changes when deletion is tested. Points and bars retain the reported estimates and 95% intervals.

Win rates average win 1, tie 0.5, loss 0 against a corrupted reference. Four + deletion has five equal components: the four original families and the mean of edge-, step-, and state-deletion tests. The compared outputs are unchanged. Methods and paired-interval protocol are those of Table 15.

F.2 What changes in the programs?

Why do the gains depend on the scoring rule? Comparing the generated structures with the supervised start reveals two different routes: reward training can add unmatched structure, while reconstruction and sample selection can remove dependencies. Table 16 tests whether either change recovers more of the required relations.

Table 16: **The same score gains correspond to different program structures.** RL and RFT alter whole outputs; X1 alters only A0 dependencies; SEL8 chooses among A0 samples. Panel (a) counts program structure, panel (b) separates coverage from conditional correctness, and panel (c) shows the selection pool. Bold cells mark the structural and relation changes that explain the score differences.

(a) Program counts per item.

Object	items	steps	direct edges	padding pairs
RL against the original verifier: whole outputs change; A0 is the start				
A0	333	11.81	17.77	33.3
P6	349	15.62	21.64	95.3
P6 − A0	331	+3.64*	+3.65*	+**59.2***
L0	352	14.64	20.73	72.0
L0 − A0	331	+2.69*	+2.79*	+36.7*
Dependency re-encoding: A0 outputs, only the dependency field changes				
X1	333	11.81	16.28	33.3
X1 − A0	333	0	−**1.48***	0
Selection-then-SFT: whole outputs change; targets re-encoded as X1				
RFT-1	336	11.39	15.96	25.0
RFT-1 − A0	328	−0.34	−**1.80***	−6.7*
RFT-2	343	11.28	15.33	24.5
RFT-2 − A0	327	−0.51*	−**2.41***	−8.1*
RFT-new	353	10.64	14.69	17.6
RFT-new − A0	332	−1.19*	−**2.87***	−16.6*
Reference-free structural selection among eight A0 samples				
SEL8	359	11.22	16.12	28.3
SEL8 − A0	332	−0.81*	−1.93*	−8.3*

(b) Relation readouts per item.

Object	items	closure pairs	ref.-ordered recovered	pair coverage	cond. correctness	ORD3
RL against the original verifier						
A0	333	50.0	31.1	0.362	0.898	0.465
P6	349	44.7	28.9	0.283	0.709	0.411
P6 − A0	331	−5.9*	−**1.82***	—	−**0.095*** (328)	−0.048*
L0	352	48.6	31.2	0.324	0.767	0.434
L0 − A0	331	−2.0	+0.44	—	−**0.047*** (328)	−0.022*
Dependency re-encoding						
X1	333	40.5	28.6	—	—	0.443
X1 − A0	333	−9.5*	−**2.45***	—	—	−0.022*
Selection-then-SFT						
RFT-1	336	38.7	29.3	0.395	0.857	0.449
RFT-1 − A0	328	−11.5*	−2.31*	—	−**0.094*** (325)	−0.022*
RFT-2	343	36.6	27.9	0.390	0.846	0.437
RFT-2 − A0	327	−14.0*	−3.00*	—	−**0.102*** (324)	−0.023*
RFT-new	353	34.9	26.9	0.397	0.854	0.431
RFT-new − A0	332	−14.6*	−3.28*	—	−**0.105*** (329)	−0.022*
Candidate selection without training						
SEL8	359	41.7	26.4	0.341	0.885	0.414
SEL8 − A0	332	−9.8*	−**4.21***	—	—	−0.042*

(c) Candidate selection: SEL8's pool of eight A0 samples per test item.

Stage	programs	steps per reference step	unmatched predicted steps
A0 greedy decode (for reference)	333 parse-valid of 360	1.056	3.12
pool, all eight samples	2,842 readable of 2,880	1.087	3.31
selected by the structural rule	360	**1.009**	3.05
the other seven	2,482	**1.098**	3.35

Steps and direct edges are per item. Padding pairs are unmatched pairs in the ORD3 denominator; closure pairs are comparable pairs in the transitive closure of written dependencies; recovered pairs follow the reference order. Coverage and conditional correctness are paired over items that express the relation. Stars mark intervals excluding zero. Panel (c) is reference-free selection among eight A0 samples and is separate from the trained outputs above.

Original-reward training increases direct edges while recovering fewer required order relations. Selected-sample fine-tuning produces sparser graphs and lower correctness on expressed relations. Across all five trained variants, these different structural changes accompany a higher original score and lower relational accuracy. The shared mechanism is a mismatch between the structure rewarded by the scorer and the relations required by the task.

The candidate pool makes the selection effect visible before training. The structural selector prefers programs closer to reference length than the other samples, yet its outputs recover fewer required ordered pairs than the greedy baseline. Length is useful for detecting padding, but cannot determine whether the retained structure binds the task correctly.

The desk example in Figure 14 explains why even edge count is insufficient. Reconstruction adds provider and protection links while removing two constraints that unnecessarily order independent actions. Correct independence is restored despite an increase in direct edges. This local repair contrasts with the test-wide loss of recovered relations in Table 16: the criterion must judge which relations are retained, rather than reward sparsity itself.

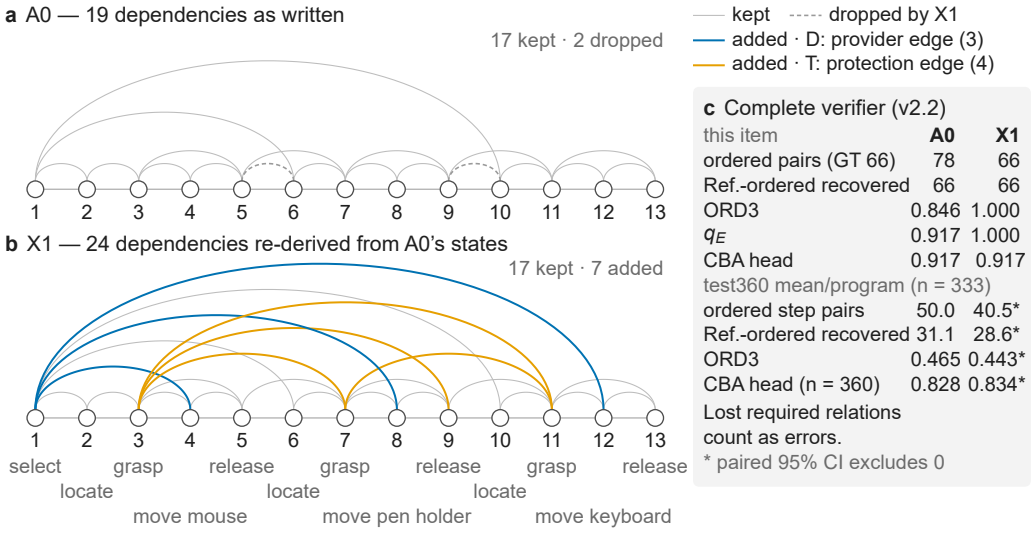


Figure 14: **Reconstructing one program's dependencies.** X1 preserves steps and states, retaining 17 edges, dropping two, and adding seven (three provider, four protection). More direct edges recover correct incomparabilities. Right: this item versus test means; stars denote paired 95% intervals excluding zero.

G Learning Interventions and Design Analyses

G.1 Reward Replacement

Starting from the same jointly supervised model, we hold the optimizer and training data fixed, replace the reward, and compare both runs after 32 updates. The key test is whether the resulting plans execute better under an engine that supplies its own operator rules and reference initial state (Table 17). This tests a behavioral consequence beyond the score being optimized.

The complete reward reduces unmatched structure and improves engine legality over both original-reward training and the supervised start. The legality gain also holds on jointly parseable programs. Relative to the original reward, fewer direct edges accompany more ordered pairs in the dependency closure and higher order agreement, showing how the intervention reorganizes action dependencies.

Relation readouts distinguish this execution improvement from a uniform score increase (Figure 15). Direct order agreement improves, whereas preference against an order-corrupted reference moves differently. The two tests ask different questions: how accurately a program represents required relations, and which of two contracts it resembles more closely. The engine comparison is the direct evidence that the reward changes execution behavior.

Table 17: **Reward replacement changes program structure and execution.** A0 is the jointly supervised start; the two endpoints use the original and complete rewards. Rows are grouped by task readouts, written structure, declared-state replay, engine execution, and goal attainment. Differences are paired with A0 or the original endpoint; intervals use the item-level bootstrap specified in Appendix D.2. The same reward endpoints are evaluated under declared states and under the independent symbolic engine.

Readout (unit)	SFT start	Original reward	Complete reward	Δ vs. SFT [95% CI]	Δ vs. original [95% CI]
(a) Task readouts *All 360 items unless marked parsed*					
Parse-valid share	0.925	0.969	0.989	$+0.0639^{*}$ [+0.0389, +0.0917]	$+0.0194^{*}$ [+0.0056, +0.0361]
ORD3 (parsed)	0.4648	0.4109	0.4663	$+0.0171$ [−0.0058, +0.0416]	$+0.0605^{*}$ [+0.0411, +0.0806]
SAT: required-edge satisfaction (parsed)	0.6178	0.7235	0.5753	-0.0267 [−0.0573, +0.0034]	-0.1430^{*} [−0.1693, −0.1171]
CBA, held-out families	0.8277	0.8457	0.8334	$+0.0057$ [−0.0131, +0.0248]	-0.0122 [−0.0333, +0.0088]
CBA, four families	0.8150	0.8349	0.8233	$+0.0083$ [−0.0047, +0.0221]	-0.0115 [−0.0245, +0.0024]
CBA, four + deletion families	0.7633	0.7900	0.7655	$+0.0022$ [−0.0114, +0.0162]	-0.0245^{*} [−0.0374, −0.0106]
CBA, step-deletion family alone	0.654	0.729	0.589	-0.0653^{*} [−0.1097, −0.0208]	-0.1403^{*} [−0.1833, −0.1000]
Signed direct score $S = 2M_{\mathrm{cc}} - 1$	−0.1863	−0.1587	−0.1375	$+0.0488^{*}$ [+0.0219, +0.0767]	$+0.0212^{*}$ [+0.0001, +0.0432]
(b) Program behaviour *Parsed: 333 / 349 / 356; paired: 333 / 348*					
Padding pairs per item (step pairs)	33.3	95.3	19.7	$\mathbf{-14.94}^{*}$ [−22.14, −8.67]	$\mathbf{-76.27}^{*}$ [−96.55, −58.44]
Steps per reference step (ratio)	1.056	1.417	0.983	-0.0796^{*} [−0.1174, −0.0434]	-0.4379^{*} [−0.5184, −0.3639]
Dependency edges per program	17.77	21.64	15.66	—	—
Comparable pairs in predicted closure	50.01	44.68	48.52	-1.70 [−6.49, +2.77]	$+3.70^{*}$ [+0.80, +6.65]

Execution legality and goal achievement.

Readout (unit)	SFT start	Original reward	Complete reward	Δ vs. SFT [95% CI]	Δ vs. original [95% CI]
(c) Declared-state replay *Submitted and sampled orders*					
Submitted order replays legally (parsed)	0.967	0.946	1.000	—	—
All 20 sampled permitted orders legal (parsed)	0.961	0.493	1.000	—	—
Mean sampled-order legal share, EXEC-1 (all items)	0.8904	0.5531	0.9889	$\mathbf{+0.0985}^{*}$ [+0.0694, +0.1290]	$\mathbf{+0.4358}^{*}$ [+0.3874, +0.4833]
(d) Engine execution (EXEC-F) *Levels: %; differences: percentage points*					
Legal share, all items	85.2	51.3	94.4	$\mathbf{+9.28}^{*}$ [+6.11, +12.57]	$\mathbf{+43.17}^{*}$ [+38.12, +48.12]
Legal share, parsed items (n 333 / 349 / 356)	92.1	52.9	95.5	$\mathbf{+3.13}^{*}$ [+0.90, +5.50]	$\mathbf{+42.64}^{*}$ [+37.72, +47.55]
(e) Goal achievement *Parsed items*					
Core goal predicates achieved	0.6456	0.6218	0.6320	-0.0120 [−0.0390, +0.0150]	$+0.0115$ [−0.0086, +0.0316]

Bold marks the paired changes in padding and execution legality. SAT measures required edges satisfied by $\hat{L}$. All scores use the commitment-complete protocol (v2.2); S is the ungated signed head mean, not the optimized reward. Padding counts step pairs with an unmatched endpoint. Reference means are 11.8 steps, 18.6 edges, and 58.8 closure pairs. Parsed rows use 333/349/356 items and paired rows use 333/348 items; all-item rows use 360. EXEC-1 averages 20 declared-state orders and EXEC-F averages 50 engine orders from the reference initial state. Goal attainment is a separate final-predicate readout. $*$: interval excludes zero; —: no paired interval recorded.

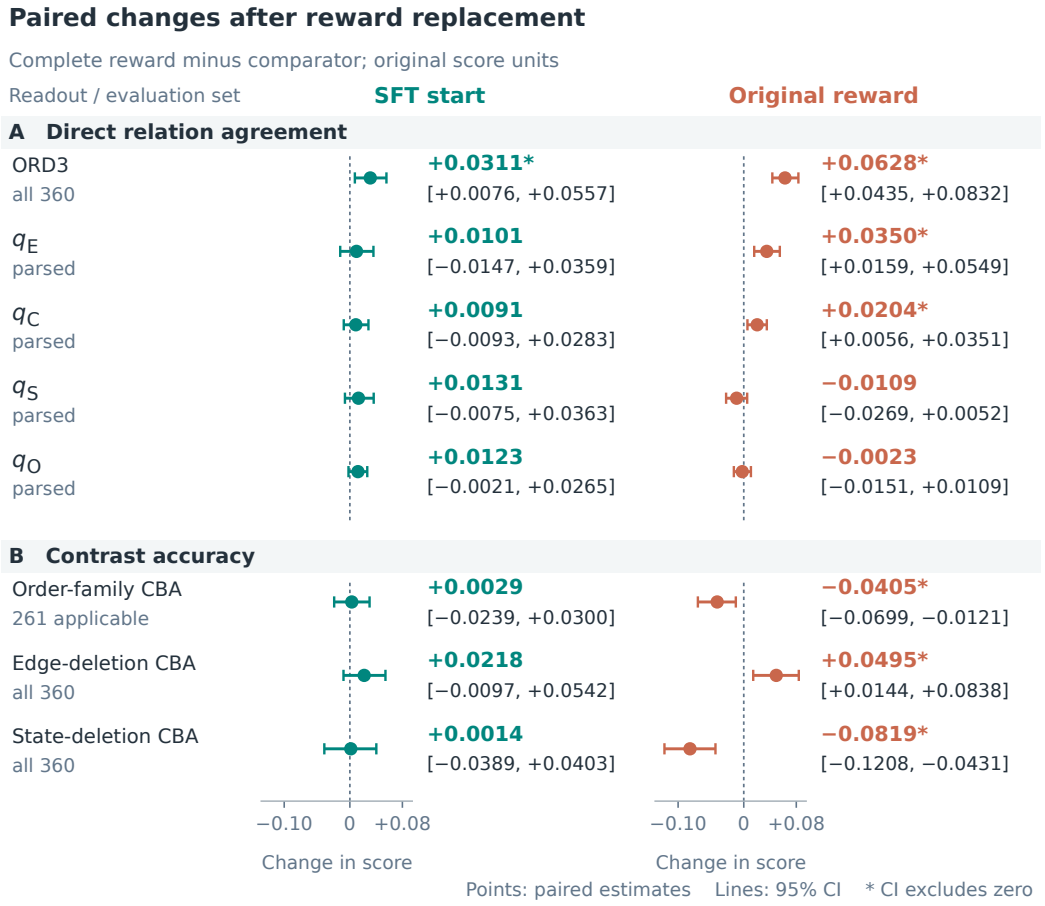


Figure 15: **Direct agreement and contrast accuracy change differently.** Complete reward is compared with the supervised start (left) and original-reward endpoint (right) after 32 updates. Points and lines show paired changes and their 95% intervals; exact values appear beside each interval. Panel A measures agreement with required relations; panel B measures preference over corrupted references. All changes retain the original score units and the listed evaluation sets. All-item ORD3 assigns parse failure 0; order-family and deletion CBA assign 0.5. Order-family CBA uses the tolerance and head floor; deletion uses the strict rule. An asterisk means the interval excludes zero.

Other training seeds. We repeat the reward comparison with data-shuffling and rollout seeds 7 and 123, retaining the shared supervised initialization, training data, and optimization recipe. Each run uses a fixed budget of 32 updates, with validation every eight updates and the final checkpoint saved and evaluated. In both comparisons, the complete reward improves symbolic execution legality and reduces unmatched step pairs (Table 18), reproducing the direction of the seed-42 result.

Table 18: **The reward effect holds across other training seeds.** Execution legality averages 50 sampled orders per task over all 360 test tasks, with parse failures scored zero. Changes are complete minus original reward. Unmatched-pair changes use jointly parseable tasks: 354 for seed 7 and 353 for seed 123. Bold marks legality gains and reductions in unmatched structure.

Seed	Original legality (%)	Complete legality (%)	Legality gain (percentage points)	Change in unmatched pairs/program
7	53.49	94.44	**+40.96**	**−39.64**
123	75.21	92.78	**+17.57**	**−45.35**

G.2 Accuracy on shared relations

Coverage and correctness answer different questions. Here the expressed relation set is fixed: 9,841 reference step pairs matched by the complete-reward output and a same-batch A0 decode across 328 tasks. Both endpoints are therefore judged on the same relations.

Table 19: **Agreement on the same relations under two weighting rules.** The same 9,841 reference pairs support both rows. Differences are complete reward minus the same-batch A0 control, in percentage points; 95% intervals resample the 328 tasks 10,000 times (seed 0).

Aggregation	A0 (%)	Complete (%)	Difference [95% CI]
Each task equally weighted	87.83	90.04	$\mathbf{+2.20}$ $[+0.47, +4.00]$
All shared pairs pooled	92.34	92.96	$+0.62$ $[-0.31, +1.58]$

Bold marks the task-equal relation change used in the main comparison; the pooled row shows how pair weighting changes the estimate.

With the evaluated relation set held fixed, complete-reward training improves accuracy when tasks are weighted equally. Pooling weights tasks by their number of shared pairs and yields a smaller estimate whose confidence interval includes zero.

G.3 Reward design

The design comparisons reveal two ways a reward can miss the intended distinction (Table 20). Harmonic aggregation ties many candidates whose orders differ, suppressing a potential learning signal. Conversely, rejecting a program whenever any sampled order fails also rejects reference programs whose submitted order is legal. One choice hides relevant differences; the other penalizes permissible task structure.

Table 20: **Offline checks guide aggregation, the submitted-order witness, and excess replay risk.** The four blocks examine candidate discrimination, structural filtering, reference treatment, and score changes on fixed outputs. The reward-replacement experiment evaluates the combined design in training.

(a) Aggregation: separating candidates with different orders

Frozen A0 test pool; original heads; candidate analysis v2; 2,835 gated candidates in 360 task groups; Spearman correlations use 200 groups.

Power mean	Order-different tie rate	Illegal-swap advantage (sd)	ρ
Harmonic, $p = -1$	0.611	0.00	0.32
Arithmetic, $p = +1$	**0.000**	0.14	0.30

(b) Structural check K0: few candidates removed

Test and training pools; training includes 672 items. Changes in ρ are ≤ 0.01.

Pool	Gated candidates	Candidates removed
Test	2,835	11
Training	5,266	14

(c) Reference replay: the submitted order supplies a legal witness

Reference sets: 360 test, 84 calibration, and 200 training; $K = 20$ sampled orders.

Reference set	Submitted-order witness: passes	Any-order gate: rejects
Test	**360/360**	21/360
Calibration	**84/84**	—
Training	—	30/200

Test diagnostics: mean illegal share 0.024; 20/21 rejected references involve single-hand interleavings; zero own-order violations. The any-order gate therefore penalizes some references with a legal submitted order. — denotes a result not reported in this comparison.

(d) Reward variants: paired scoring changes on the same frozen outputs

Frozen P6 update-32 versus A0; test-360; $B = 2{,}000$. The earlier v2 heads predate ORD3.

Scoring variant	Paired ΔR	95% CI
Original heads	$+0.0798$	$[+0.0541, +0.1072]$
v2 heads	$+0.0438$	$[+0.0192, +0.0699]$
v2 with any-order gate	-0.2791	$[-0.3245, -0.2328]$
v2 with excess penalty ($\beta = 1$)	-0.2165	$[-0.2572, -0.1761]$

All four intervals exclude zero. The excess-form reference score is **1.000** on **360/360** references: it charges excess failures without penalizing reference risk.

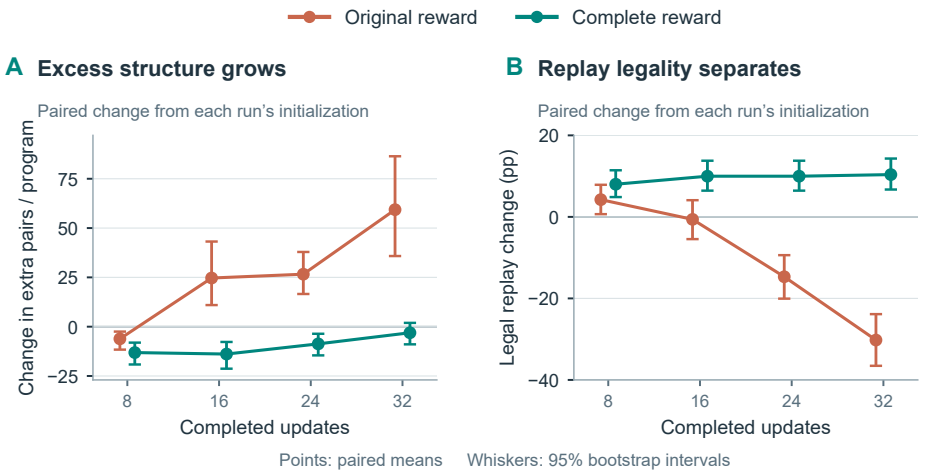
Bold identifies the design choices and their supporting comparisons. sd is the median illegal-swap advantage divided by the within-item reward standard deviation across eight candidates; ρ is the within-group Spearman correlation with the order-family contrast. Tie rate counts equal rewards among order-different candidate pairs. Reference replay follows Appendix C.1. Candidate comparisons use the original heads; the final training run uses commitment-complete v2.2 heads. These offline comparisons motivate the combined reward.

Arithmetic aggregation resolves the order-different ties, while a submitted-order witness checks the executable sequence the model actually commits to. Reference-relative replay risk then penalizes

additional failures without requiring every interleaving of a partially ordered reference to be legal. Together, these choices preserve useful order sensitivity and a neutral reference baseline; the reward-replacement experiment tests their combined effect in training.

G.4 Training dynamics

The training records locate when the two behaviors diverge (Figure 16). After an initial reduction in unmatched structure, the original reward reverses direction: extra pairs accumulate and replay legality falls. Complete-reward training sustains the legality gain while keeping program length near its starting level. Both runs improve parsing and raise the original contrast score, while their execution behavior diverges. The checkpoint trajectories connect this divergence to the accumulation of unmatched structure under the original reward.



C All five checkpoints, all eight readouts

Cell values are observed levels; shading is shared between arms within each metric.

Readout	Original reward					Complete reward				
Training update →	0	8	16	24	32	0	8	16	24	32
Extra step pairs	22.4	17.7	49.2	54.9	**85.3**	24.5	12.1	10.7	17.4	**23.2**
Physical steps	11.6	11.2	13.3	14.0	**15.4**	11.7	10.2	10.2	11.1	**11.6**
Dependency edges	18.0	16.9	19.9	20.0	**21.8**	18.2	15.8	16.0	16.7	**17.0**
Legal replay (%)	88.4	92.7	87.8	73.8	**58.2**	88.8	96.9	98.8	98.8	**99.2**
Order agreement (%)	46.2	48.9	48.9	46.9	**43.6**	47.0	45.8	45.6	48.7	**48.5**
Parsed programs (%)	89.8	94.5	98.0	98.4	**98.8**	90.6	96.9	98.8	98.8	**99.6**
ORIGINAL SCORING RULE · SELECTION READOUTS										
Held-out contrast	0.665	0.711	0.723	0.750	**0.761**	0.667	0.478	0.449	0.594	**0.692**
Order contrast	0.523	0.589	0.616	0.655	**0.670**	0.523	0.178	0.121	0.370	**0.536**

Structure counts use parsed programs; legality and agreement use all 254 tasks.

Figure 16: **Training separates execution behavior from the original selection scores.** (A,B) Changes from each run's initialization, with task-paired 95% bootstrap intervals (5,000 resamples). A uses programs parsed at both checkpoints; B uses all 254 tasks, counting failed parses as illegal. (C) All five checkpoint means: structure counts use each checkpoint's parsed programs; legality and order agreement use all 254 tasks. Behavior uses the complete scoring rule; the two contrast rows retain the original rule (254/199 applicable tasks). Shading is scaled separately for each metric, jointly across both rewards.

G.5 Distribution of program changes

The distributions distinguish changes in typical programs from changes in the tail (Figure 17). Under the original reward, a long padding tail pulls the mean beyond the median, and execution outcomes spread between fully legal and fully illegal. The complete reward contracts that tail and concentrates more programs at full legality. Pairing the same tasks shows how these effects arise: execution changes combine many unchanged tasks with a substantial group that gains legality, while padding reductions vary widely in magnitude. Appendix H follows one change through the actions themselves.



Figure 17: **Reward replacement reduces the padding tail and concentrates execution at full legality.** (A,B) Empirical histograms, individual programs, and quartiles expose the spread behind the means. Padding counts pairs with an unmatched endpoint. (C,D) Circle areas show exact frequencies of legal fractions, preserving partial outcomes and the masses at zero and one. A–D use parsed programs (supervised 333, original 349, complete 356). (E,F) Complete-minus-comparator changes use the same paired tasks (348/333 versus original/supervised); row counts distinguish decreases, ties, and increases. Padding axes use signed $\log_{10}(1 + |x|)$ spacing with labels in original units.

H ONE TASK, THREE TRAINING ENDPOINTS

Under the original reward, this task expands into many steps that cannot be matched to the reference; the complete reward returns close to the reference length (Figure 18). The action alignment links the reduction in padding to changes in the action sequence. The accompanying readouts quantify coverage, order agreement, and execution legality.

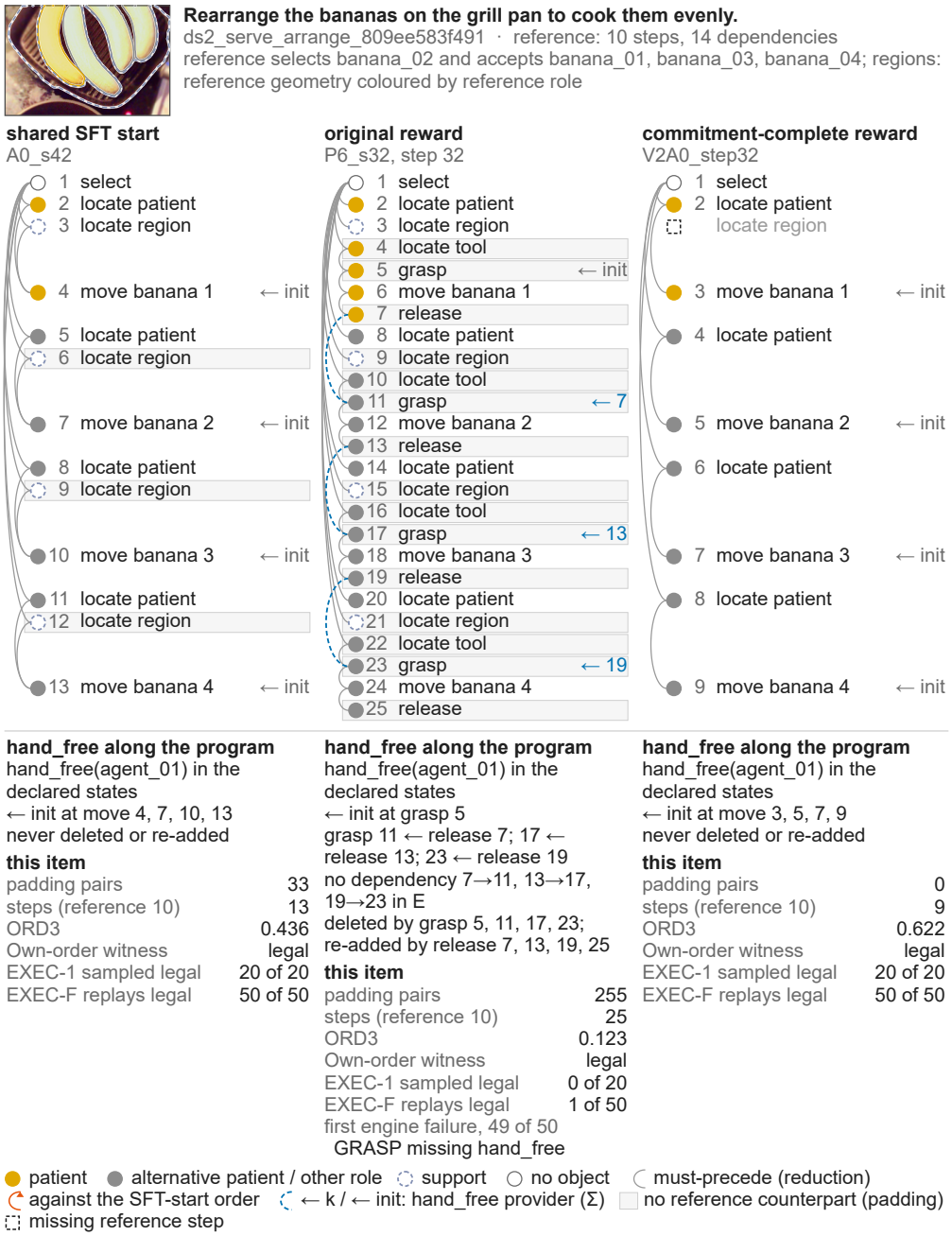


Figure 18: **The same task under three training endpoints.** The shared supervised start and the two reward endpoints are aligned with the reference. The original reward adds many unmatched steps; the complete reward returns close to the reference length. The dashed square marks one required localization omitted by that endpoint; the bottom readouts separate coverage, order agreement, and execution legality.